UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA
PROGRAMA DE PÓS–GRADUAÇÃO EM INFORMÁTICA APLICADA

IGOR VICTOR LUCENA DO NASCIMENTO

DEEPREF: UM FRAMEWORK PARA CLASSIFICAÇÃO DE RELAÇÕES BASEADO EM DEEP LEARNING

RECIFE – PE

2022

IGOR VICTOR LUCENA DO NASCIMENTO

DEEPREF: UM FRAMEWORK PARA CLASSIFICAÇÃO DE RELAÇÕES BASEADO EM DEEP LEARNING

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada do Departamento de Estatística e Informática - DEINFO - Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

ORIENTADOR: Rinaldo Lima

RECIFE – PE

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

N244d Nascimento, Igor Victor Lucena do
DeepREF: um Framework para Classificação de Relações Baseado em Deep Learning / Igor Victor Lucena do Nascimento. - 2022.
120 f. : il.

Orientador: Rinaldo Jose de Lima.
Inclui referências e apêndice(s).

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2022.

1. Classificação de Relações. 2. Framework. 3. DeepREF. 4. Embeddings. 5. Processamento de Linguagem Natural. I. Lima, Rinaldo Jose de, orient. II. Título

CDD 004

IGOR VICTOR LUCENA DO NASCIMENTO

DEEPREF: UM FRAMEWORK PARA CLASSIFICAÇÃO DE RELAÇÕES BASEADO EM DEEP LEARNING

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada do Departamento de Estatística e Informática - DEINFO - Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Aprovada em: 01 de 01 de 2022.

BANCA EXAMINADORA

Rinaldo Lima (Orientador)
Universidade Federal Rural de Pernambuco
Departamento de Computação

Filipe Rolim Cordeiro
Universidade Federal Rural de Pernambuco
Departamento de Computação

Vanilson Andre de Arruda Burégio
Universidade Federal Rural de Pernambuco
Departamento de Computação

Dedico este trabalho à minha família: esposa, filha, pais, irmãos e amigos.

Agradecimentos

Agradeço primeiramente a Deus, autor da vida e motor de todas as coisas. Depois a meus pais que cooperaram na minha criação e me incentivaram para os estudos. Agradeço a meu irmão que, por sua paixão pela tecnologia, me levou a trilhar a área de Computação. Agradeço à minha irmã que, por meio da sua linguagem, me levou a apaixonar-me pelo estudo da linguagem.

Agradeço também à minha esposa que no seu silêncio me anima a correr atrás dos meus objetivos. Agradeço a minha filha, recém-nascida, que, com o seu sorriso, me dá alegria e forças pra vencer os obstáculos da vida.

Agradeço, por fim, a todos os que colaboraram para meu crescimento pessoal, profissional e moral, sejam eles professores, amigos e colegas. Que Deus possa recompensá-los.

O saber não entra num espírito mesquinho.

(Dom Bosco)

Resumo

Extração de Relações (ER) é uma tarefa básica e importante de Processamento de Linguagem Natural (PLN) para muitas aplicações, incluindo motores de busca e sistemas de perguntas e respostas. Existem muitos estudos nesta subárea de PLN que continua a ser explorada, tais como aquelas relacionadas às tarefas do SemEval. Por muitos anos, vários sistemas ER baseados em modelos estatísticos tinham sido propostos, assim como os frameworks para desenvolvê-los. Porém, poucos são os frameworks que disponibilizam pontos de extensão em relação à otimização de hiperparâmetros, aplicação de pré-processamento de texto em datasets de dominínios diferentes e utilização de diversos tipos de embeddings de contexto. Este trabalho tem como foco os frameworks que permitem desenvolver tais sistemas ER usando modelos Deep Learning (DL). Tais frameworks tornam possível reproduzir experimentos usando muitos modelos DL e técnicas de pré-processamento de datasets. Atuamente, existem poucos frameworks deste tipo e nenhum que tenha essa essas duas funções ao mesmo tempo. Neste trabalho é proposto um framework aberto, otimizável, com facilidade de uso e extensível, tanto para modelos como para datasets, chamado DeepREF, inspirado por dois outros frameworks existentes: OpenNRE e REflex. O DeepREF permite um rápido desenvolvimento de modelos DL para Classificação de Relações (CR), devido a sua extensibilidade tanto para datasets como para modelos e sua facilidade de uso herdada do OpenNRE. Porém, o OpenNRE apenas possui o benefício de poder utilizar os modelos estado-da-arte e de criar novos com facilidade, devido ao reuso, deixando a desejar no processamento e geração de novos datasets. Além do mais, o DeepREF disponibiliza otimização de hiperparâmetros e a aplicação de várias técnicas de pré-processamento nos dados textuais de entrada, inspirado pelo REflex. DeepREF possui mais facilidade de uso de extensão de técnicas de pré-processamento do que o REflex, pois é um sistema que possui maior coesão e acoplamento. Também fornece meios de acelerar o processo de treinamento de modelos DL para tarefas de CR em diferentes datasets e modelos. DeepREF é avaliado em três diferentes datasets e tem demonstrado resultados competitivos comparados à outros sistemas CR estados-da arte.

Palavras-chave: Classificação de Relações. Framework. DeepREF. Embeddings. SemEval. DDI. Optuna, PLN

Abstract

Relation Extraction (RE) is a basic and important task of Natural Language Processing (NLP) for many applications, including search engines and question-answer systems. There are many studies in this NLP subarea that continue to be explored, such as those related to SemEval tasks. For many years, various ER systems based on statistical models had been proposed, as well as the frameworks to develop them. However, there are few frameworks that provide extension points in relation to hyperparameter optimization, application of text preprocessing in datasets from different domains and use of different types of context embeddings. This work focuses on the frameworks that allow developing such ER systems using Deep Learning (DL) models. Such frameworks make it possible to reproduce experiments using many DL models and dataset preprocessing techniques. Currently, there are few frameworks of this type and none that have these two functions at the same time. In this work, an open, optimizable, easy-to-use and extensible framework is proposed, both for models and for datasets, called DeepREF, inspired by two other existing frameworks: OpenNRE and REflex. DeepREF allows rapid development of DL models for Relation Classification (RC), due to its extensibility for both datasets and models and its ease of use inherited from OpenNRE. However, OpenNRE only has the benefit of being able to use state-of-the-art models and create new ones easily, due to reuse, leaving something to be desired in processing and generating new datasets. Furthermore, DeepREF provides hyperparameter optimization and the application of various preprocessing techniques on the input textual data, inspired by REflex. DeepREF has more ease of use and extension of pre-processing techniques than REflex, as it is a system that has greater cohesion and coupling. It also provides means to speed up the process of training DL models for CR tasks on different datasets and models. DeepREF is evaluated on three different datasets and has demonstrated competitive results compared to other state-of-the-art CR systems.

Keywords: Relation Classification. Framework. DeepREF. Embeddings. SemEval. DDI. Optuna. NLP

Lista de Figuras

Lista de tabelas

Lista de Siglas

IA	*Inteligência Artificial*
ML	*Machine Learning*
DL	*Deep Learning*
PLN	*Processamento de Linguagem Natural*
EI	*Extração de Informação*
CR	*Classificação de Relações*
EN	*Entidades Nomeadas*
NER	*Named Entity Recognition*
ER	*Extração de Relações*
CNN	*Convolution Neural Network*
RNN	*Recurrent Neural Network*
GNN	*Graph Neural Network*
GRU	*Gated Recurrent Unit*
LSTM	*Long Short-Term Memory*

Sumário

1 Introdução

No mundo moderno, cada dia mais aumenta o número de dados na Internet seja por redes sociais ou por sites, artigos de notícias, blogs, publicações e fóruns gerando uma grande quantidade de informações todos os dias. Com isso, tem-se tornado impossível analisar manualmente todos os dados gerados em mídias digitais. Uma área da Inteligência Artificial (IA) chamado de Processamento de Linguagem Natural (PLN) busca tratar esses problemas por meio do entendimento de textos com linguagens utlizadas por seres humanos para extrair informações úteis de forma automática.

A Extração de Informação (EI), um tópico de PLN, lida com o problema de obter informações estruturadas a partir de textos. Existem amplamente 2 estágios em um sistema de EI: primeiro, a identificação de entidades ou conceitos e; segundo, a identificação de relações existentes entre essas entidades. O foco da presente dissertação é a identificação das relações existentes entre duas entidades (relações binárias), conhecida como Classificação de Relações (CR) na literatura.

Muito se tem estudado sobre a classificação de relações e tem-se obtido muito bons resultados. Porém, existem dificuldades na reprodução desses experimentos por outros pesquisadores da área de PLN devido a falta de clareza e informações sobre detalhes, seja da arquitetura do modelo de rede neural utilizado para a classificação de relações, seja dos parâmetros encontrados para a obtenção dos resultados apresentados nas pesquisas. Além disso, não existe uma padronização nos formatos da base de dados dos diferentes domínios disponíveis para experimentos pelos pesquisadores. Isso dificulta bastante a comparação entre diferentes abordagens do mesmo problema utilizando diferentes modelos de arquitetura e diferentes formas de processamento da base de dados para os experimentos (CHAUHAN et al., 2019).

Os *frameworks* e ferramentas existentes que auxiliam na reprodutibilidade de modelos de arquitetura DL utilizando-se diferentes configurações de hiperparâmetros e dataset é o *OpenNRE* e o *REflex*. O *OpenNRE* possui uma grande capacidade de modularização e reuso de modelos de arquiteturas para CR estados-da-arte. Já o *REflex* possui módulos de pré-processamento de datasets (a fim de padronizar o formato de diversos datasets existentes), otimização e avaliação de modelos. Essas duas principais características do *OpenNRE* e *REflex*, respectivamente, ajudam a resolver o problema da

reprodutibilidade.

Existem diversos modelos de arquitetura para CR que inicialmente foram utilizados como comparação em alguns datasets mas não foram reproduzidos em outros datasets e que poderiam ter resultados promissores. Há duas dificuldades principais para experimentação de um modelo de arquitetura de *Deep Learning* (DL) em outros tipos de datasets: primeiro, existe a dificuldade de reproduzir o modelo de arquitetura proposto em alguns artigos por diversas razões: falta de código fonte, omissão de recursos de variabilidade de detalhes de hiperparâmetros, falta de informações importantes sobre detalhes de arquitetura de modelo; segundo, existem diferentes formatos de datasets para serem processados e, assim, serem usados para o treinamento de modelos DL, o que dificulta e retarda o processo de experimentação de diferentes datasets. No presente trabalho, foram utilizados datasets de diferentes domínios, tais como dominínios geral, científico e biomédico.

A área biomédica torna-se uma área potencial de pesquisa para o uso de técnicas de mineração de dados e extração de informações, pois é cada dia mais crescente o número de informações biomédicas de publicações em artigos, tais como PUBMED e MEDLINE. O número de artigos publicados na área é cada vez mais crescente e torna-se manualmente impossível obter as informações mais recentes de determinado problema biomédico, dificultando os profissionais da saúde na busca de novos tratamentos mais eficazes e eficientes de pacientes. Dessa forma, o presente trabalho busca focar na extração de informações, especificamente na classificação de relações de base de dados do domínio da área biomédica e científica. Também foram realizados experimentos na base de dados de domínio geral como meio de comparação com os resultados da base de domínio biomédico e científico.

Um exemplo de aplicação real da classificação de relações é a procura artigos da área biomédica em busca das novas descobertas de relações entre medicamentos (que é o objetivo do dataset DDI Extraction 2013 (HERRERO-ZAZO et al., 2013)) em tempo real. Esse tipo de aplicação pode auxiliar os profissionais da área de saúde a estarem atualizados, em tempo hábil, quanto aos novos experimentos de interação entre medicamentos, evitando complicações em pacientes devido ao uso indevido de diferentes medicamentos. Isso ocorre por conta do crescimento exponencial da quantidade de artigos que são publicados na área biomédica. Esse crescimento exponencial de publicações da área biomédica de novos experimentos de interação entre medicamentos torna a atualização do conhecimento por

parte dos profissionais de saúde praticamente impossível através de um esforço puramente humano.

Portanto, o objetivo da presente pesquisa é a união dos objetivos dos dois *frameworks* citados acima: *OpenNRE* e *REflex*. O *OpenNRE* busca o balanceamento do encapsulamento do sistema, eficiência operacional, extensibilidade do modelo e facilidade de uso (HAN et al., 2019). Enquanto que o *REflex* desenvolve um *framework* que possibilita a comparação de várias metodologias de treinamento, pré-processamento, técnicas de modelagem e métricas de avaliação. Tudo isso para identificar melhores práticas de classificação de relações e servir como guia para abordagens de novos datasets (CHAUHAN et al., 2019). Em suma, essa dissertação objetiva a criação de um framework para classificação de relações binárias baseado em DL que preencham as lacunas de usabilidade, customização e otimização das soluções chamado de *DeepREF*. O *DeepREF* é destinado, *a priori*, aos pesquisadores de PLN, que tenham algum conhecimento de programação em *Python* e no *framework* de *machine learning* chamado *Pytorch*, focados na classificação de relações binárias de entidades possibilitando, mais facilmente, comparações mais claras e evidentes das melhorias que as diferentes abordagens apresentam para a classificação de relações de dados de domínio em geral. Dessa forma, é possível extrair conclusões sobre melhorias nos resultados ao utilizar o *framework* que permite ser extendido para outros tipos de domínios e *datasets*.

1.1 Objetivos

A seguir serão apresentados os objetivos geral e específicos que nortearão a condução desta pesquisa.

1.1.1 Objetivo Geral

A presente pesquisa tem como objetivo geral um mecanismo fácil e rápido que permita reproduzir o desempenho de diferentes modelos de arquiteturas de Deep Learning (DL) publicados na literatura para Classificação de Relações (CR) e otimizar esse mesmo desempenho dos modelos por meio de técnicas de otimização de hiperparâmetros, aplicando-os a datasets de diferentes domínios e utilizando-se de diferentes técnicas de

pré-processamento de datasets e de diferentes tipos de *embeddings* de contexto. Tudo isso atrelado a facilidade de uso do *framework* por meio de encapsulamento do sistema permitindo o reuso e extensão de funcionalidades no *framework*.

1.1.2 Objetivos Específicos

Para atingir o objetivo geral, foram propostos os seguintes objetivos específicos:

- Implementação de um novo *framework* a partir de *frameworks* já existentes;
- Estruturação do *framework* com classes e relações utilizando-se princípios de padrões de projeto para facilitar o uso e extensão das técnicas de pré-processamento de datasets;
- Escolha do modelo com maior desempenho e experimentação com diferentes configurações de hiperparâmetros, *embeddings* e técnicas pré-processamento de dataset;
- Melhoria da arquitetura de modelo, alterando-se a estrutura da rede neural de forma a atingir o melhor desempenho;
- Seleção do modelo com maior desempenho e otimização dos hiperparâmetros por meio de um *framework* de otimização de hiperparâmetros;
- Comparação do modelo com melhor desempenho no *framework* do presente trabalho com outros *frameworks* e ferramentas que possuem o mesmo objetivo e modelos estado-da-arte.

1.2 Contribuições

As principais contribuições desta dissetação foram:

- o desenvolvimento de um *framework* que permita a otimização de desempenho e criação de diferentes modelos de arquitetura de redes neurais artificiais utilizando-se *datasets* de diferentes formatos de forma mais eficiente;
- o *benchmarking* de diferentes modelos com hiperparâmetros otimizados e combinação de diferentes técnicas de pré-processamentos de *datasets*;
- criação de um modelo estado-da-arte para a maioria dos datasets investigados neste trabalho.

1.3 Escopo Negativo

Segue o escopo negativo do presente trabalho. O presente trabalho não comtempla:

- a classificação de relações entre mais de duas entidades (n-árias);
- a classificação de relações entre entidades de sentenças vizinhas (*ER* bag-level), em um documento (*DocRE*) ou *few-shot ER*;
- o *framework* como serviço na Web, mas apenas como *framework* para desenvolvedores de *frameworks* de *machine learning*, tais como *Pytorch* e *Tensorflow*;
- a versão do presente *framework* para o *Tensorflow*, mas apenas para o *Pytorch*.

1.4 Justificativa

Dado o sucesso de modelos baseados em DL aplicados à extração e classificação de relações, mostrando uma forte habilidade e atingindo altas *performances*, houve um crescimento cada vez maior da atenção de pesquisadores e desenvolvedores da indústria neste campo de estudo. Mesmo os modelos de DL aplicados à CR serem efetivos e serem aplicados em vários cenários, faltam ferramentas efetivas e estáveis que suportem a implementação, *deployment* e avaliação de modelos voltados para classificação de relações (HAN et al., 2019).

Reprodutibilidade é muito importante para validação de trabalhos anteriores e a construção de novos modelos. A falta de reprodutibilidade na área de PLN ocorre devido a vários fatores, tais como, a não disponibilidade do código-fonte do modelo de arquitetura na *Internet* e a omissão de fontes de variabilidade, tais como os detalhes dos hiperparâmetros. Além disso, existe a falta de modularidade que permite que o código-fonte seja facilmente extensível a novos modelos de arquitetura e *datasets*. Numa pesquisa feita por (CHAUHAN et al., 2019), 16 dos 53 artigos relevantes encontrados sobre classificação de relação utilizando redes neurais artificiais tinham disponibilizado o código-fonte. 14 dos 53 artigos fizeram exeprimentos em vários *datasets*, mas apenas 5 desses disponibilizaram o código-fonte. Mesmo que muitos artigos tenham disponibilizado detalhes importantes de hiperparâmetros, alguns outros hiperparâmetros importantes não foram disponibilizados, tais como, número de épocas, tamanho do *batch* e *seed* de inicialização randômica. Outros detalhes importantes de técnicas não foram informados nos artigos como, por exemplo, o uso da técnica de *early stop*.

A falta de reprodutibilidade causa também uma avaliação empírica desapropriada para identificar as fontes de ganhos na modelagem das arquiteturas de redes neurais, bem como a forma de pré-processamento dos dados. Um estudo de ablação é um estudo que compara a *performance* de um sistema de IA, ou modelos de arquitetura de redes neurais, por meio da remoção de certos componentes para entender a contribuição de cada componenente no sistema como um todo. (CHAUHAN et al., 2019) pesquisou que, entre os 53 artigos pesquisados, 20 dos 24 artigos do domínio geral fez estudos de ablação. Por outro lado, apenas 10 dos 29 artigos no domínio médico realizaram um estudo de ablação. Entre esses estudos de ablação, faltaram detalhes importantes relacionados ao pré-processamento que foram críticos nos experimentos da referida pesquisa.

Dessa forma, o presente trabalho visa a construção e disponiblização do *DeepREF*: um framework *open-source* que permite realizar um *benchmarking* de diferentes modelos de arquitetura, pré-processamento, técnicas de modelagem, otimização de hiperparâmetros e métricas de avaliação. O código-fonte do *DeepREF* está disponível em <https://github.com/igorvlnascimento/DeepREF>. No repositório do código existe um passo-a-passo explicando como utilizar o *framework*. Além de disponibilizar o *framework*, foram realizados experimentos e estudos de ablação que comprovam a eficácia do referido *framework*.

2 Fundamentação teórica

Este capítulo desenvolve conceitos importantes para o entendimento do trabalho realizado tais como tarefas de PLN (Seção 2.1), Extração e Classificação de Relações (Seção 2.2), Redes Neurais Artificiais (Seção 2.3), *Deep Learning* (Seção 2.4), conceito e tipos de *embeddings* (Seção 2.5), otimização de hiperparâmetros (Seção 2.6) e conceito e vantagens de frameworks (Seção 2.7).

2.1 Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN) é uma subárea da computação linguística que analisa linguagens naturais de forma automática. As aplicações dessa tecnologia inclui *Machine Translation* (MT), sumarização automática, extrações de informações, entre outras.

O PLN é uma área bastante desafiadora por 2 razões: as linguagens naturais tem bastante ambiguidade, e as palavras podem ser combinadas numa sentença de diferentes formas, tornando impossível, para computadores, simplesmente listar os possíveis contextos e significados de uma palavra numa sentença (LIMA; FREITAS, 2014).

2.1.1 Análise morfológica e sintática

O PLN nas aplicações de mineração de texto tipicamente consiste de aplicações sequenciais de vários componentes que realizam subtarefas de PLN (LIMA; FREITAS, 2014). Normalmente, as subtarefas de PLN são realizadas sequencialmente como mostra a figura 1. Cada subtarefa será explicada abaixo.

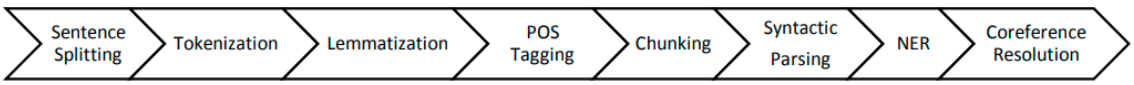

Figura 1 – Uma forma sequencial comum de serem realizadas as subtarefas de PLN, da esquerda para a direita (LIMA; FREITAS, 2014)

Sentence splitting consiste em, a partir de um texto bruto, retornar as sentenças

separadas umas das outras.

Tokenization é o processo de quebrar o texto ou a sentença em palavras, símbolos ou outros elementos significativos de um texto, referidos como *tokens*. Nem todos os espaços em branco indicam a fronteira de um *token*. Existem palavras que são tidas mais expressões ou nomes próprios como "*Nova York*", por exemplo.

Lemmatization consiste no processo de reduzir cada palavra em sua forma base, ou *lemma*. Considere o verbo "amar" que pode ser conjugado como "amarei", "amo", "amas", "amando" e seu *lemma* é "*amar*", que é a sua forma básica de todas as suas formas flexionadas.

Part-of-Speech (POS) tagging é a tarefa de atribuir para cada *token* o seu correspondente *part-of-speech tag*, isto é, suas categorias de palavras sintáticas, tais como, substantivo, adjetivo, verbo, entre outros (LIMA; FREITAS, 2014). A figura 2 mostra os resultados da extração de POS tags para a sentença: "*The quick brown fox jumped over the lazy dog*".

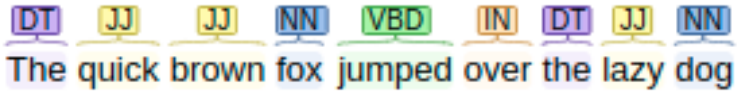

Figura 2 – Resultados do POS tagging para a sentença "*The quick brown fox jumped over the lazy dog*"[1].

A tabela 1 mostra o significado de cada POS *tag* mostrada na figura 2. Existe um conjunto de *POS tags* proposto pela *Penn Treebank Project* que foi um projeto de um grupo de pesquisadores que desenvolveram um corpus com 4,5 milhões de palavras do Inglês Americano anotados com informações de *POS tags* (MARCUS et al., 1993). Eles geraram um conjunto de *POS tags* para anotar as palavras que ainda é usado por outros pesquisadores.

Chunking consiste em dividir a sentença em grupos de palavras correlacionadas sintaticamente, como substantivos, verbos, e frases preposicionais. Em outras palavras, *chunks* são grupos de palavras que formam uma pequena unidade sintática.

Na língua Inglesa, a principal unidade (*head*) em uma frase substantiva é comumente o substantivo mais à direita. Por exemplo, na frase substantiva "*the exciting modern art museum*", a palavra "*museum*" denota a principal unidade constituinte, enquanto as outras palavras estão modificando ou restringindo o significado do substantivo *head*. A grande

Tag	Descrição
DT	Determinante
JJ	Adjetivo
NN	Substantivo singular
VBD	Verbo no passado
IN	Preprosição

Tabela 1 – Descrições dos *POS tags* da figura 2

vantagem do *chunking* é sua robusteza e eficiência (LIMA; FREITAS, 2014).

***Syntactic analysis* (*parsing*)**, ao contrário de *chunking*, se trata da análise e construção de estruturas sintáticas de acordo com uma gramática, que é uma formalidade que descreve as estruturas sintáticas corretas numa linguagem.

Um algoritmo de *parsing* é um procedimento de busca de possíveis modos de combinar regras gramaticais para encontrar um ou mais estruturas que combinam com uma dada estrutura de sentença.

Existem dois tipos de gramáticas que surgiram para analisar e gerar árvores *parsing* em linguagem natural:

- *Gramática de estrutura de frases.* Define estruturas em termos de categorias frasais, como por exemplo, *noun phrases* (NP), *verb phrase* (VP), entre outros. A figura 3 mostra um exemplo de árvore de análise produzido por um analisador de estrutura de frase para a sentença "*Black swans like clear ponds*" (LIMA; FREITAS, 2014).
- *Gramática de dependência.* A teoria dessa gramática formaliza a construção de uma estrutura de dependência, ou grafo de dependência entre 2 unidades linguísticas que dominam imediatamente uma a outra na árvore de sintaxe. Análise de dependência realiza uma análise sintática completa de sentenças de acordo com a teoria da gramática de dependência (LIMA; FREITAS, 2014). A figura 4 mostra um exemplo de grafo de dependência para uma sentença em Inglês, onde as relações de dependência são representadas por arestas dirigidas começando da palavra *head* apontando a palavra *dependente*, que é a palavra subordinada sintaticamente ao *head*. A tabela 2 possui informações sobre os rótulos das dependências da figura

4 e suas respectivas descrições. Existe um *treebank*[2], como o *Peen Treebank* para os *POS tags*, com um conjunto de dependências em várias línguas chamado de *Universal Dependencies*[3] criado para facilitar as pesquisas em análise de dependências multilíngue (MCDONALD et al., 2013).

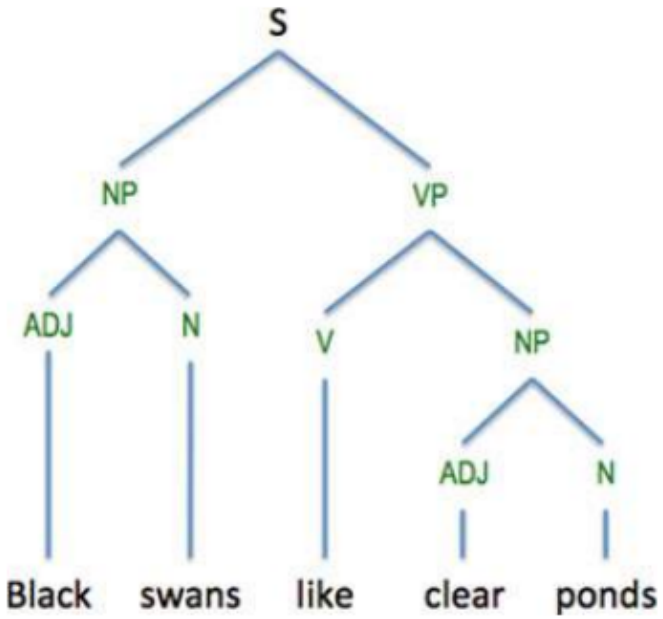

Figura 3 – Estrutura de frase ou análise constituinte de uma sentença em Inglês (KAMATH et al., 2019) .

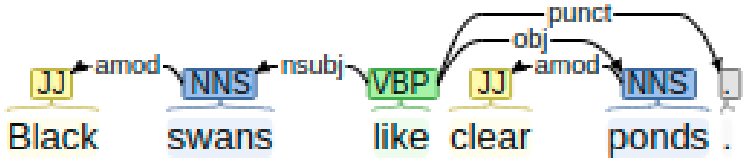

Figura 4 – Grafo de dependência de uma sentença em Inglês.

***Named Entity Recognition* (NER)** é a tarefa de processamento de linguagem natural responsável por identificar e nomear palavras ou frases num texto que se refere a entidades como pessoas (PER), lugar (LOC), organização (ORG), tempo, ou quantidade (MISC) (KAMATH et al., 2019; SANG; MEULDER, 2003). Ela também é considerada uma subtarefa de EI e algumas vezes chamada de extração de entidades. *Hidden Markov Models* (HMM) e *Conditional random fields* (CRFs) têm tido algum sucesso em NER

[2] Corpus de texto que tem sido analisado e anotado para estruturas sintáticas
[3] <https://universaldependencies.org/>

Rótulo	Descrição
amod	Adjetivo modificador
nsubj	Sujeito nominal
obj	Objeto direto
punct	Pontuação

Tabela 2 – Descrições dos tipos de dependência (MCDONALD et al., 2013)

(SANG; MEULDER, 2003). Porém, treinar esses tipos de modelos requer uma grande quantidade de dados de treinamento anotado. Mesmo com muitos dados, NER ainda não é, de forma geral, uma tarefa resolvida (KAMATH et al., 2019).

Coreference resolution é um processo linguístico em que os anáforas são ligados com seus antecedentes. Quando ocorre dentro de um documento é chamado de *local coreference*. Se ocorre entre vários documentos, é chamado de *global coreference*. Pode ser considerado uma tarefa de classificação (KAMATH et al., 2019). A figura 5 mostra um exemplo de *coreference resolution* em que a palavra "*her*" se refere à "*Mary*" na sentença: "*Mary is going to Brazil. John is going with her*".

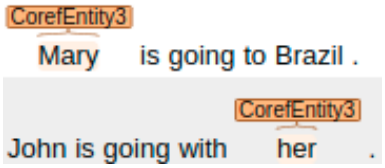

Figura 5 – Exemplo de *coreference resolution*.

2.2 Extração e Classificação de Relações

A EI é o processo de extrair informações e transformar essas informações em dados estruturados, ou também a atividade de popular uma fonte de conhecimento estrutrado com a informação de um texto não-estruturado (GAIZAUSKAS; WILKS, 1998). Esse texto estruturado pode ser usado para outras tarefas, tais como mineração de texto. Existem vários tipos de informações que podem ser extraídas, como por exemplo nomes próprios e as relações existentes entre eles.

Todo texto não trivial possui um grupo de entidades ou conceitos e interações

e interrelacionamentos entre si. Identificar essas entidades e as relações entre si é um passo fundamental para o entendimento do texto. *Named Entity Recognition* (NER) apresentam problemas bem estudados em PLN com uma vasta pesquisa na literatura. NER é a identificação dos *tokens* que correspondem à menções de entidades. Esse passo do reconhecimento de entidades é anterior à ER ou CR (NASTASE et al., 2013). O foco desse presente trabalho é a etapa de CR.

Assim como um leitor humano, um sistema PLN deve explorar conhecimento pré-adquirido sobre o mundo e novas informações dadas pelo texto. O primeiro recurso é frequentemente construído como uma base de conhecimento, e seu conteúdo pode ser fixo ou dinâmico à medida que mais conteúdo é lido (NASTASE et al., 2013).

Relações são as conexões percebidas entre conceitos ou entidades. Mineração de relações em um texto tem um grande potencial em extrair relações de entidades em um grande banco de dados automaticamente em bem menos tempo do que se fosse realizado manualmente. Artigos biomédicos, por exemplo, crescem de forma exponencial e é impossível que um pesquisador biomédico rastreie tudo o que é publicado. A habilidade de identificar relações semânticas num texto pode ser útil em várias tarefas de PLN, tais como: tradução de máquina (*machine translation*), EI, sumarização de texto, entre outros (NASTASE et al., 2013).

O objetivo essencial da EI é extrair um tipo específico de informação de um dado repositório de documento e colocar isso como saída para um repositório estruturado, tal como uma tabela relacional ou um arquivo *XML*. Envolve processamento de linguagem natural, computação linguística e mineração de texto. A EI também é útil em *question-answering*, recuperação de informações, entre outras coisas (PAWAR et al., 2017).

As informações de um texto que se quer extrair são: Entidades Nomeadas (EN), relações e eventos. O foco desse trabalho são nas relações entre ENs. Uma EN é, na maioria das vezes, uma palavra ou frase que representa um objeto específico no mundo real. NER é a tarefa para identificar todas as ocorrências de uma particular EN num documento de texto (PAWAR et al., 2017).

A ER é definida como a tarefa de extrair relações semânticas entre argumentos. Os argumentos podem ser conceitos gerais tais como "uma empresa" (ORG), "uma pessoa" (PER) ou instâncias de tais conceitos, i.e. "*Apple*" e "*Steve Jobs*", os quais são chamados de EN (BACH; BADASKAR, 2007).

A definição formal para a ER é dada por um corpus de texto C que contém n sentenças $C = (s_1, ..., s_n)$. Cada sentença s_i possui uma tripla $t_i = (e1, r_i, e2)$, onde *e1* e *e2* são as EN e r_i é a relação entre as entidades. Deve-se obter um classificador *f*, usando sentenças anotadas heuristicamente como um conjunto de treinamento, tal que recebe como entrada uma nova sentença anotada s_i com duas EN e retorna a relação entre tais instâncias expressas em r_i (ASSIS, 2015).

As relações a serem extraídas numa sentença podem ser binárias, terciárias e assim por diante (n-árias). O presente trabalho focará nas relações binárias. Uma relação binária consiste num sujeito ou entidade-cabeça, i.e. "*Steve Jobs*", um predicado, i.e. "fundador-de" e um objeto ou entidade-cauda, i.e. "*Apple*" (AUGENSTEIN et al., 2014). Elas são representadas pela tripla $(e1, r_i, e2)$, respectivamente.

Atualmente, existem muitos modelos de DL que apresentam bons resultados nas tarefas de CR. Esses modelos ultrapassam no desempenho dos resultados em relação à modelos de *machine learning* tradicionais.

2.3 Redes Neurais Artificiais

Redes Neurais Artificiais é um subconjunto de *Machine Learning* e é parte essencial nos algoritmos de *Deep Learning*. Seu nome é inspirado no cérebro humano, e tenta reproduzir a forma como os neurônios biológicos enviam sinais um para o outro (EDUCATION, 2020).

A forma mais simples de uma rede neural é representada por um *Perceptron*. O Perceptron é um modelo matemático que recebe várias entradas e produz uma única saída binária. Um *Perceptron* simples com apenas 1 neurônio é representado pela figura 6. Ele foi desenvolvido pelo cientista Frank Rosenblatt entre as décadas de 1950 e 1960 (ACADEMY, 2022). A saída do neurônio do *Perceptron* é 0 ou 1 e é determinado se a soma ponderada for maior ou menor do que algum limiar, como na equação 2.1.

$$output = \begin{cases} 0, & \text{if } \sum_j w_i x_j \leq threshold \\ 1, & \text{otherwise} \end{cases} \tag{2.1}$$

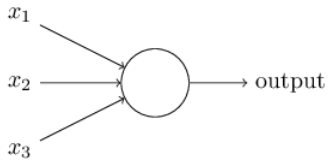

Figura 6 – Exemplo de um *perceptron* simples.

Com o passar do tempo e aumento do poder de processamento dos computadores, foi possível executar *perceptrons* mais robustos com mais neurônios e multi-camadas, o *Multi-Layer Perceptron* (MLP). O MLP apresenta camadas escondidas que ficam entre a camada de entrada e de saída como mostrado na figura 7. O *Deep Learning*, que será visto na próxima seção, começa quando há redes neurais cada vez mais profundas como num MLP, por exemplo.

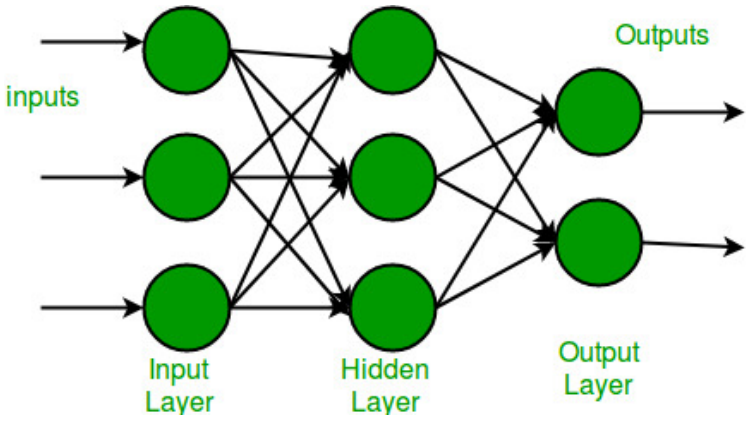

Figura 7 – Exemplo de um MLP.

2.4 Deep Learning

Vive-se na era do *big data* em que uma enome quantidade de dados são gerados todos os dias. Com isso, há uma urgente necessidade de analisar e extrair informações desses dados. Os algoritmos de *Machine Learning* (ML) podem ajudar. *Deep Learning* (DL) é um subconjunto de ML, que é um subconjunto da IA. A DL é inspirado no processamento de padrões de informações existentes no cérebro humano. DL não precisa de um conjunto de regras humanas para operar, mas necessita de uma enorme quantidade de dados que mapeam a entrada para o resultado específico. DL é projetado usando vários

camadas de algoritmos (redes neurais artificiais). Ao contrário do DL, as técnicas de ML convencionais precisam de vários passos sequenciais (pré-processamento, extração de *features*, seleção, aprendizagem e classificação) para se chegar na tarefa de classificação. Diferente de métodos de ML cconvencionais, DL tem a habilidade de automatizar o aprendizado de um conjunto de características para várias tarefas (ALZUBAIDI et al., 2021).

DL ainda está em contínuo desenvolvimento e tem trazido melhorias para muitos campos de aprendizado, tais como: super-resolução de imagens, detecção de objetos e reconhecimento de imagens. Recentemente, a performance de DL tem superado a performance de humanos em tarefas como classificação de imagens. Dessa forma, DL pode ser uma solução para casos em que epecialistas não são disponíveis, casos em que o problema é sempre atualizado o tempo todo, ou casos em que o problema possui um numero muito grande de dados, tornando impossível, humanamente, a análise de todos os dados, em pouco tempo e, assim, tomar alguma decisão (ALZUBAIDI et al., 2021).

2.4.1 Classificação de abordagens de DL

Modelos de DL pode ser treinados utilizando-se 3 tipos de abordagens principais, tais como: supervisionado, semi-supervisionado e não-supervisionado.

A abordagem supervisionada trata de dados anotados, ou seja, dado um conjunto de entradas, cada uma delas terá uma saída correspondente ao valor esperado de cada entrada. O papel do modelo DL é fazer uma predição do valor dado um conjunto de entradas. No treinamento, é calculado um valor de *loss* que exprime o quão longe o valor da predição está do valor esperado. A partir desse valor de *loss*, o modelo se ajusta para que função retorne o valor mais próximo dos valores esperados. Ou seja, o valor de *loss* deve tender a zero que significa o que os valores preditos e esperados estão mais próximos entre si (ALZUBAIDI et al., 2021).

Existe também a abordagem do aprendizado semi-supervisionado em que o *dataset* não possui todos os dados anotados, mas apenas alguns. Uma das vantagens dessa técnica é minimizar a quantidade de dados anotados necessária. Por outro lado, a desvantagem é de que dados irrelevantes podem estar presentes nos dados de treino e isso pode levar o modelo a tomar decisões incorretas. Classificador de textos de documentos é a aplicação mais

utilizada com essa abordagem por conta da dificuldade em obter uma grande quantidade de dados de texto de documentos anotados (ALZUBAIDI et al., 2021).

Por fim, a abordagem de aprendizado não-supervisionado torna possível o processo de aprendizado sem a presença de dados anotados. Nessa abordagem, o modelo busca características internas do próprio dado e suas relações entre si a fim de descobrir estruturas e relacionamentos não identificados. A principal desvantangem do aprendizado não-supervisionado é que ele é incapaz de fornecer informações precisas sobre classificação de dados e computação complexa (ALZUBAIDI et al., 2021).

2.4.2 Tipos de redes DL

Um dos tipos mais famosos de redes DL são: *Convolution Neural Networks* (CNNs), *Recurrent Neural Networks* (RNNs), *Graph Neural Network* (GNNs) e *Transformers*. Cada uma delas será aprofundada nessa seção, assim como também serão mostradas as principais aplicações dessas redes.

2.4.2.1 Convolution Neural Networks

As CNNs tem sido o suporte principal de modelos para tarefas de visão computacional (ASIF et al., 2021). As arquiteturas completamente convolucionais utilizam uma operação de convolução para representação do aprendizado e utilizam camadas altamente conectadas achatando o vetor resultante em apenas uma dimensão (ASIF et al., 2021). A figura 8 mostra um exemplo de arquitetura da CNN para classificação de imagens. Cada *pixel* da imagem a ser treinada na CNN é uma entrada para a camada de entrada. Os nós da camada de Convolução podem ser representados por $t \times n_2 \times 1$, e $n_2 = n_1 - k_1$, onde n_1 é o número de canais de entrada da CNN e k_1 é o tamanho do *kernel* utilizado para as convoluções (ASIF et al., 2021). O mapeamento de ativação da camada de convolucional pode ser obtido como:

$$y^{i(p)} = max(0, b^{j(p)}) + \sum_j k^{ij(p)} * x^{i(p)} \tag{2.2}$$

onde $x^{i(p)}$ e $y^{i(p)}$ são definidos como a i-ésima entrada e a j-ésima saída do mapeamento de ativação, respectivamente. $b^{j(p)}$ é o viés do j-ésimo mapeamento da saída e $*$ representa a convolução. $k^{ij(p)}$ é representado como o *kernel* da convolução entre

a i-ésima entrada e a j-ésima saída (ASIF et al., 2021). A figura 9 mostra como ocorre a convolução de forma gráfica.

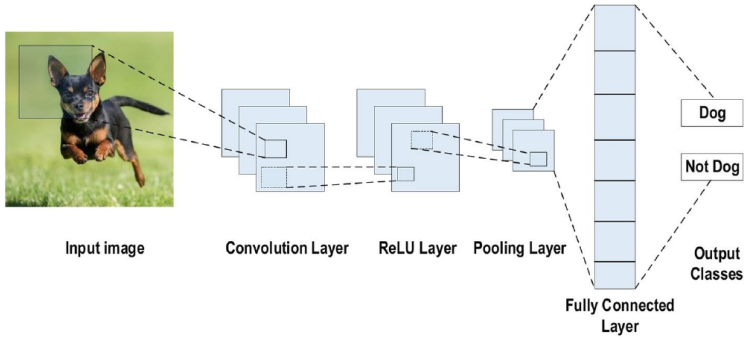

Figura 8 – Um exemplo de arquitetura CNN para classificação de imagens (ALZUBAIDI et al., 2021)

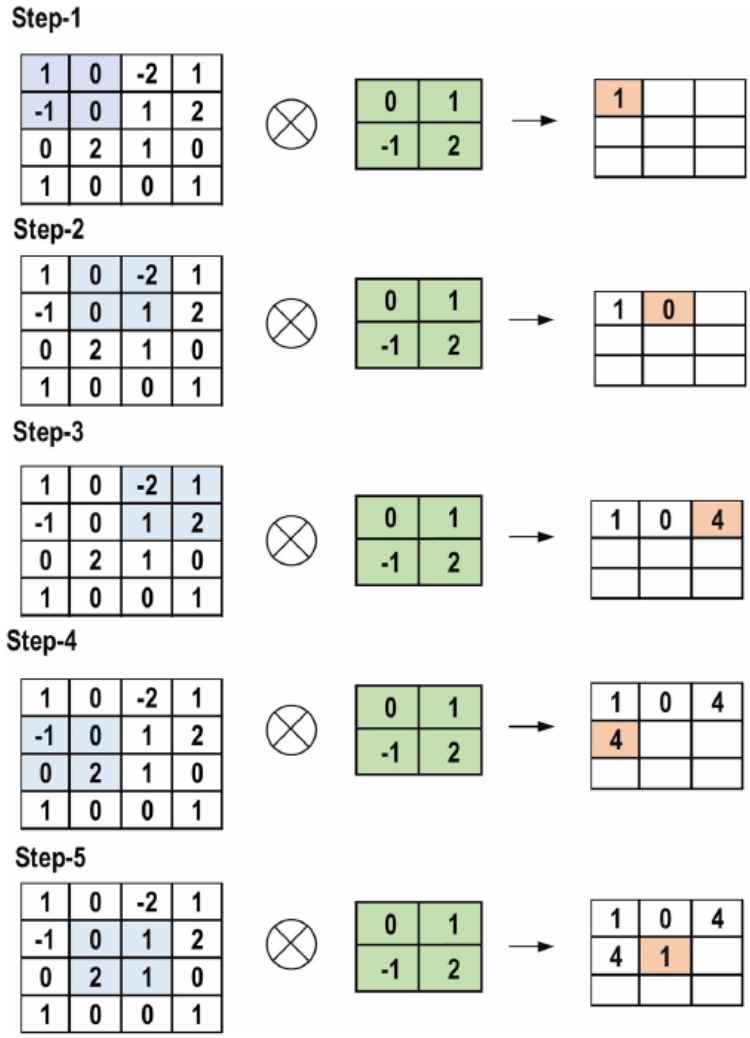

Figura 9 – Os principais cálculos executados em cada passo por uma camada de convolução (ALZUBAIDI et al., 2021)

A arquitetura CNN consiste de várias camadas. Cada camada convolucional, que é o componente mais significante da arquitetura, contém um grupo de filtros convolucionais (também chamados de *kernels*). A imagem de entrada é convolucionada por esses filtros como mostra a figura 9. O *kernel* é uma grade de números discretos em que cada valor é o peso do *kernel* (ALZUBAIDI et al., 2021).

Em algumas arquiteturas CNN existem as camadas de *pooling* que servem para diminuir o tamanho grande dos mapeamentos de *features* gerados pela camada de convolução e também serve para manter as imformações mais relevantes em cada estágio de *pooling*. Existem vários tipos de métodos *pooling* e os mais usados são o *max*, *min* e GAP (*Global Average Pooling*). A figura 10 mostra os tipos de *pooling* mais usados nas CNNs.

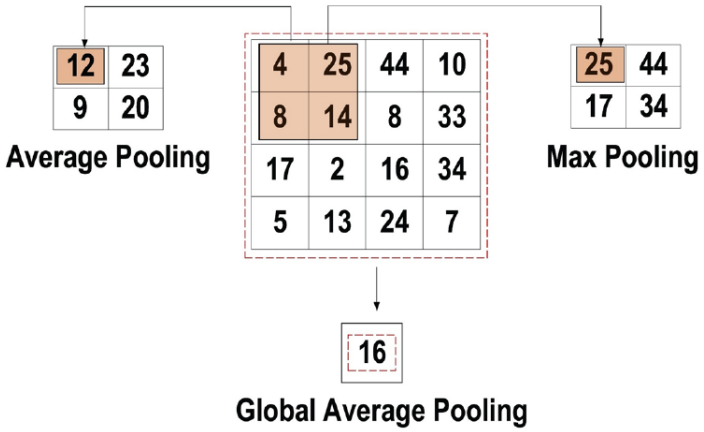

Figura 10 – Os tipos de operação de *pooling* (ALZUBAIDI et al., 2021)

2.4.2.2 Recurrent Neural Networks

RNNs são frequentemente aplicados na área de reconhecimento de voz e em contextos PLN. Esse tipo de rede utiliza dados sequenciais. Isso é muito importante no contexto de interpretação de textos, pois o entendimento do significado de uma palavra em um texto se dá pelo conhecimento do contexto. Dessa forma, é possível entender RNNs como unidades de memória de curto-prazo. Porém, um dos maiores problemas das RNNs são os *exploding e vanishing gradients* (ALZUBAIDI et al., 2021). Esse problema ocorre pois numa RNN acontece uma sequência de multiplicações desde a primeira palavra de um texto até a última, fazendo com que os valores das primeiras palavras decresçam, pois multiplica-se

números muito pequenos que tendem a zero. Com isso, palavras importantes vão perdendo a importância ao decorrer das sequências na RNNs. Por exemplo, na sentença "Maria vai para o Brasil e João vai com ela" é fácil identificar que o pronome "ela" se refere à "Maria". Mas nas RNNs o nome "Maria" perde a importância ao longo dos cálculos e perde-se a informação da relação existente entre "Maria" e "ela". Para lidar com esse problema, foram desenvolvidos extensões das RNNs como GRUs e LSTMs. Essas redes possuem em cada bloco de memória contém células de memória que armazenam os estados temporais da rede. Também possuem unidades de portas que controlam o fluxo de informações, diminuindo, assim, o problema do *exploding e vanishing gradients* (ALZUBAIDI et al., 2021).

O *Long Short-Term Memory* (LSTM) foi desenvolvido por (HOCHREITER; SCHMIDHUBER, 1997) com a ideia de manter uma informação longa por um certo período de tempo. LSTMs aproveitam o armazenamento de memória na forma de funções de ativações de termos curtos. (ASIF et al., 2021).

O nó de entrada recebe a ativação da entrada da sequência de entrada $x^{(t)}$ no passo de tempo atual e o passo de tempo corrente é processado pelo passo de tempo anterior $h^{(t)}$ que são chamados de estados escondidos (*hidden states*, em inglês). Geralmente, a soma ponderada é calculada usando a função de ativação chamada tangente hiperbólica (*tanh*), mas o artigo principal usa a função de ativação *sigmoid*. A expressão vetorizada do estado interno do bloco de memória é dado como $s^{(t)} = g^{(t)} \cdot i^{(t)} + s^{(t-1)}$, onde $\cdot$ é a multiplicação por cada elemento. O trabalho de (GERS; SCHMIDHUBER, 2000) acrescentou o *forget gate* que é uma técnica de decidir sobre a deleção do conteúdo do estado interno, caso necessário. O *forget gate* fará esse controle de fluxo. Isso faz com que o fluxo do gradiente ao longo do LSTM se torne mais suave (ASIF et al., 2021). A expressão do estado interno com o *forget gate* pode ser descrita como:

$$s^{(t)} = g^{(t)} \odot i^{(t)} + f^{(t)} \odot s^{(t-1)} \tag{2.3}$$

onde $g^{(t)}$ é o *output gate*, $i^{(t)}$ é o *input gate* e $f^{(t)}$ é o *forget gate*.

2.4.2.3 Graph Neural Networks

O trabalho de (SCARSELLI et al., 2009) foi que primeiro introduziu a GNN aproveitando as redes neurais para exploração de dados no domínio de grafos. A GNN visa a predição de anotações de dados não-anotados (ASIF et al., 2021). Matematicamente, a GNN pode ser representada como:

$$h_v = f(y_v, y_{co[v]}, h_{ne[v]}, y_{ne[v]}) \tag{2.4}$$

onde $y_{co[v]}$ define as representações de arestas ligadas ao nó v e $h_{ne[v]}$ e $y_{ne[v]}$ define os *embeddings* e *features* dos nós vizinhos ao nó v, respectivamente. A função f define a função de transição que transforma o vetor de entrada em um espaço dimensional d (ASIF et al., 2021). A equação 2.4 entra num processo iterativo de atualização que pode ser expressado como:

$$H^{t+1} = F(H^t, Y) \tag{2.5}$$

Esse processo é conhecido como *message passing* ou *neighborhood aggregation*. Dessa forma, a função de saída da GNN é expressada como:

$$o_v = G(h_v, y_v) \tag{2.6}$$

As funções $F()$ e $G()$ são redes *feed-forward* altamente conectadas.

As GNNs mostraram uma significante perfomance utilizando estratégias digiridas a dados estruturados em grafos. Apesar do bom desempenho, esse método não aprende informações longas e foca em todos os nós igualmente. Assim, muitos nós irrelevantes são também considerados para a tarefa desejada. Além disso, GNN tem um viés indutivo arbitrário, que causa uma generalização fraca enquanto aplica-o em outros tipos de dados como vídeos e imagens (ASIF et al., 2021).

Existem outros tipos de GNNs e umas das mais conhecidas são a *Graph Convolution Network* (GCN) e a *Graph Attention Network* (GAN).

A GCN faz um esforço de combinar as redes de grafos com as CNNs a fim de explorar o potencial das CNNs e aplicar modelos voltados à grafos. A GCN é baseada na

regra de propagação por camada. Utiliza também operações de convolução para explorar dados estruturados em grafos (ASIF et al., 2021).

A GAN funciona em problemas tanto indutivos como transdutivos aproveitando o potencial da unidade de atenção para aprender diferentes regiões de espaço de *feature* utilizando o mecanismo de auto-atenção. Como o mecanismo de auto-atenção é computacionalmente custoso, esse método teraá um desempenho melhor em dados de grafos em larga escala (ASIF et al., 2021).

2.4.2.4 Transformers

O *Transformer* mostra que os mecanismos de atenção são muito eficientes e precisam de menos computação em relação a outros modelos. O *Transformer* é construído inteiramente sobre mecanismos de atenção e alcança estado-da-arte em tarefas de PLN, tais como tradução de línguas. Um dos mecanismo de atenção usados no *Transformer* é chamado de auto-atenção, também chamado de intra-atenção, e está relacionada a diferentes posições de uma única sequência em ordem para computar uma representação de uma sequência. Auto-atenção tem sido usado também para outras tarefas de PLN, tais como: compreensão de leitura, resumo abstrativo, vinculação textual e aprendizado de representações de sentenças independentes da tarefa (VASWANI et al., 2017).

A figura 11 mostra a arquitetura do *Transformer* que usa auto-atenção empilhadas com camadas altamente conectadas para ambos *encoder* e *decoder*.

O uso de *Transformers* foi melhorado em abordagens baseadas em *fine-tuning* e, assim, originou-se o *Bidirectional Encoder Representations from Transformers* (BERT). O BERT utiliza um "modelo de linguagem mascarado" (MLM) como objetivo de pré-treino. Esse modelo mascara aleatoriamente alguns dos *tokens* de entrada, e o objetivo é a predição do *id* do vocabulário original da palavra mascarada baseado apenas no seu contexto. O objetivo do modelo MLM habilita a fusão da representação do contexto da direita e esquerda, permitindo o pré-treino de um *Transformer* bidirecional profundo, que é o BERT. O BERT pode ser ajustado com apenas uma camada adicional de saída para criar modelos estados-da-arte em uma gama de tarefas. O BERT é conceitualmente simples mas empiricamente poderoso. Ele alcança estados-da-arte em 11 tarefas de PLN (DEVLIN et al., 2019).

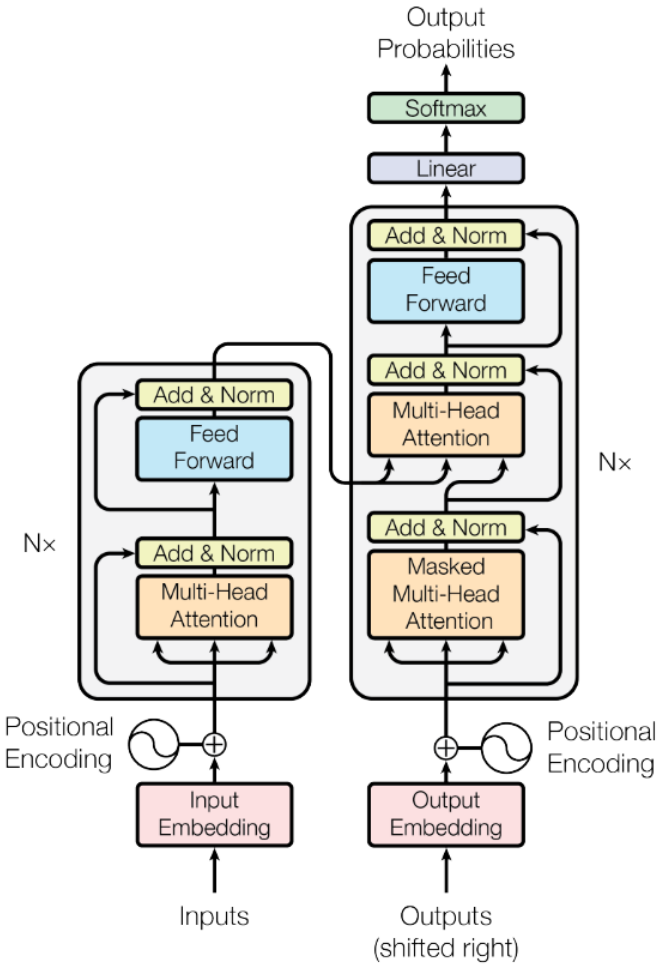

Figura 11 – Modelo de arquitetura do *Transformer*.

Durante o desenvolvimento do BERT, foram escolhidos alguns hiperparâmetros, tais como o número de camadas de auto-atenção, o número de camadas ou blocos do Transformer e o tamanho das camadas escondidas. Foram construídos BERTs de 2 tamanhos por (DEVLIN et al., 2019) que são o BERT *base* e o BERT *large*. O BERT *base* contém 12 blocos do Transformer, 768 camadas escondidas e 12 camadas de auto-atenção, resultando num total de 110 milhões de parâmetros. Este possui o mesmo tamanho do modelo *Generative Pre-trained Transformer* (GPT) de (RADFORD et al., 2018), para fins de comparação. O BERT *large* com 24 blocos do Transformer, 1024 camadas escondidas e 16 camadas de auto-atenção, resultando num total de 340 milhões de parâmetros. A performance desses dois tipos de BERT ultrapassam outros modelos, porém o BERT *large* é melhor do que o BERT *base* (DEVLIN et al., 2019). No presente trabalho, utiliza-se o BERT *base* pois este é mais rápido e menor do que o BERT *large*. Para fins de comparação, utiliza-se apenas o BERT *base* para todos os experimentos que utiliza-se o BERT.

Houve outros trabalhos que otimizaram o pré-treino do BERT originando outras versões do BERT, tais como o RoBERTa (LIU et al., 2019), o XLNet (YANG et al., 2019), o DistilBERT (SANH et al., 2019), que é uma versão menor e mais leve do BERT, entre outros. A tabela 3 mostra um pouco as características dos diferentes tipos de BERT.

	BERT	**RoBERTa**	**DistilBERT**	**XLNet**
Tamanho (em milhões)	**Base:** 110 **Large:** 340	**Base:** 110 **Large:** 340	**Base:** 66	**Base:** 110 **Large:** 340
Tempo de treinamento	**Base:** 8 x V100 x 12 dias, **Large:** 64 TPU Chips x 4 dias	**Large:** 1024 x V100 x 1 dia (4-5 vezes mais do que o BERT)	**Base:** 8 x V100 x 3,5 dias (4 vezes menos tempo do que o BERT)	**Large:** 512 TPU Chips x 2,5 dias (5 vezes mais do que o BERT)
Performance	Supera os estados-da-arte em outubro de 2018	2-20% de melhoria em relação ao BERT	3% de piora em relação ao BERT	2-15% de melhoria em relação ao BERT
Dados	16 GB de dados BERT (*corpus* de livros + Wikipedia). 3,3 bilhões de palavras.	160 GB (16 GB de dados do BERT + 144 GB de dados adicionais)	16 GB de dados do BERT	**Base:** 16 GB de dados do BERT **Large:** 113 GB (16 GB de dados do BERT + 97 GB de dados adicionais). 33 bilhões de palavras.

Tabela 3 – Comparativo dos diferentes tipos de BERT

Existem outros tipos de BERT pré-treinados em um domínio específico como de textos e resumos de artigos científicos como o SciBERT (BELTAGY et al., 2019) e de domínio biomédico como o BioBERT (LEE et al., 2019). O presente trabalho utiliza o SciBERT nos experimentos com os modelos neurais no domínio biomédico e cientifico, pois o SciBERT apresenta melhores resultados para tarefas de classificação de relações com o dataset *ChemProt* (BELTAGY et al., 2019) que é de domínio biomédico (KRALLINGER et al., 2017).

2.4.3 Funções de ativação

A camada de função de ativação (não-linear) toma a decisão de se deve ou não disparar um neurônio com referência a uma entrada específica criando a saída correspondente (ALZUBAIDI et al., 2021). Camadas de ativação não-lineares são empregadas depois de todas as camadas com pesos, tais como as camadas altamente conectadas e camadas convolucionais. A não-linearidade da camada de ativação signifca que o mapeamento da entrada e saída será não-linear. A função de ativação também tem a habilidade de diferenciar, o qual é uma característica extremamente significante, pois permite o processo de *back-propagation* dos erros a serem usados para treinar a rede. Existem vários tipos de função de ativação e as mais usadas são a *sigmoid* (equação 2.7), a tangente hiperbólica (*Tanh* - equação 2.8) e a *ReLU* (equação 2.9). A entrada da função da *sigmoid* são números reais, enquanto a saída é restrita entre 0 e 1. A *sigmoid* é representada graficamente por uma curva em S. A tangente hiperbólica é similar à função *sigmoid*, tendo sua entrada como números reais e como saída é restrita entre 1 e -1. A *ReLU* é a função mais comumente utilizada pois converte os valores negativos em 0. O baixo processamento computacional da *ReLU* é o principal benefício dela sobre as outras funções de ativação (ALZUBAIDI et al., 2021).

$$sigmoid = \frac{1}{1 + e^{-x}} \tag{2.7}$$

$$tahn = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.8}$$

$$ReLU = max(0, x) \tag{2.9}$$

2.5 Embeddings

Embeddings tem sido uma das mais influentes área de pesquisa em PLN. Um *embedding* é uma informação codificada para representação de vetores de baixa dimensão que é facilmente integráveis em modelos de *machine learning* modernos. As técnicas de *embedding* inicialmente focaram-se em palavras, mas com o tempo foram-se

utilizando outras formas, tais como estruturas de grafos para bases de conhecimento, e outros tipos de conteúdo textual, tais como sentenças e documentos (PILEHVAR; CAMACHO-COLLADOS, 2020).

2.5.1 Word embeddings

Word embeddings são uma forma de representação distribucional para palavras em um vocabulário, onde cada palavra é representada por um vetor de baixa dimensão espacial (baixa em relação ao tamanho do vocabulário). Os *word embeddings* visam capturar informações sintáticas e semânticas sobre a palavra e são obtidos por meio de métodos não-supervisionados utilizando-se um enorme *corpus* de texto não anotado. São implementados usando uma matriz de *embedding* $E \in \mathbb{R}^{|V| \times d_w}$, onde d_w é a dimensionalidade do espaço do *embedding* e $|V|$ é o tamanho do vocabulário (KUMAR, 2017).

Por meio da hipótese distribucional (FIRTH, 1957), que diz que palavras que ocorrem no mesmo contexto tendem a ter significados semelhantes, e do advento das DL, surgiram muitas técnicas de obtenção dos *word embeddings* tais como o *Word2Vec* (MIKOLOV et al., 2013) que, apesar de não ser "deep", é um modelo muito eficiente para construção de vetores compactos (PILEHVAR; CAMACHO-COLLADOS, 2020).

Outro método de *word embedding* é o *Glove* (PENNINGTON et al., 2014) que é uma técnica baseada em fatorização matricial de matrizes de contexto de palavras. Existe também uma melhoria do *Word2Vec* que é o *FastText* (BOJANOWSKI et al., 2016), que inclui caracteres *n-grams* permitindo o cálculo de representação de palavras que não estão no vocabulário.

Existe também o *Embeddings from Language Models (ELMo)* que é uma representação de palavra profundamente contextualizada (profundamente no sentido de que se utilizou DL para geração dos *word embeddings*). O *ELMo* modela as características complexas de uma palavra, como a sintaxe e semântica, e a forma como os usos da palavra varia entre contexto linguísticos (PETERS et al., 2018).

Os *Transformers* também geram *word embeddings*. Exemplos de *word embeddings* utilizando os *Transformers* são aqueles já treinados como o BERT, RoBERTa e o XLNet. Eles são explicitados na seção 2.4.2.4. Os *word embeddings* gerados pelos *Transformers*

são mais robustos e possuem um conhecimento mais profundo da palavra em comparação com os *word embeddings* explicados anteriormente.

2.5.2 Outros embeddings

Existem outros tipos de *embedding* diferente dos *word embeddings*. Um deles são os *positional embeddings* (ZENG et al., 2014). que são capturados ao obter as distâncias relativas entre os *tokens* de uma sentença às entidades de uma sentença que se quer extrair a relação. Nos *positional embedding* gerados por (ZENG et al., 2014), existem dois vetores que serão os *positional embeddings* da entidade cabeça e cauda, respectivamente.

Outros embeddings com outras informações da sentença como os *POS tags* e os grafos de dependência nos quais podem ser pré-treinados ou serem apenas injetados nos modelos de treinamento de forma aleatória para serem treinados juntamente com o problema que se quer resolver.

2.6 Otimização de Hiperparâmetros

Otimização de hiperparâmetros (OHP) é o problema de otimizar a função de custo sobre um espaço de configuração com estrutura de grafos. A otimização de hiperparâmetros não apenas otimiza variáveis discretas ou contínuas mas, ao mesmo tempo, escolhe quais variáveis otimizar. Otimizar os hiperparâmetros maximiza o desempenho do modelo no conjunto de validação (ELGELDAWI et al., 2021).

Existem vários tipos de algoritmos de OHP, tais como a busca em grade, busca randômica, otimização bayesiana e estimadores de *Tree Parzen*.

2.6.1 Busca em grade

A busca em grade é o mais básico método para OHP. Ele realiza uma busca exaustiva no conjunto de hiperparâmetros definidos pelos usuários. O usuários devem ter conhecimento prévio dos hiperparâmetros porque são eles quem geram todos os candidatos. A busca em grade é aplicada para vários hiperparâmetros com espaço de busca limitado (YU; ZHU, 2020).

Mais especificamente, a otimização de busca em grade gera um produto cartesiano de todas as possíveis combinações de hiperparâmetros. Esse método simplesmente constrói uma grade com cada combinação possível de todos os valores de hiperparâmetros fornecidos, calculando as pontuações de cada modelo e, então, seleciona os modelos com os melhores resultados. Para executar a busca em grade deve-se selecionar um conjunto finito de valores razoáveis para cada hiperparâmetro. O algoritmo de busca em grade treina o modelo com cada combinação de hiperparâmetros no produto cartesiano. Depois de todos esses cálculos, o algoritmo de busca em grade retorna as configurações que atingem o melhor desempenho no procedimento de validação. O melhor conjunto de hiperparâmetros escolhido pela busca em grade é usado no modelo atual. A vantagem da busca em grade é que esse método garante a detecção dos melhores hiperparâmetros, porém sua complexidade algorítmica cresce exponencialmente numa taxa de $O(n^k)$, sendo n o número dos distintos valores testados e k o número de parâmetros (ELGELDAWI et al., 2021).

2.6.2 Busca randômica

A busca randômica é uma melhoria básica da busca em grade. A busca randômica indica uma busca randomizada ao longo dos hiperparâmetros de uma certa distribuição sobre possíveis valores de parâmetros. Esse método de otimização de hiperparâmetros é similar à busca em grade mas mostrou-se criar melhores resultados especialmente quando alguns hiperparâmetros não são uniformemente distribuídos. A busca randômica é mais suscetível a rapidamente encontrar uma configuração ótima do que a busca em grade. Porém, não é certeza obter uma configuração ótima. É mais certo que quanto maior o tempo da busca randômica, maior a probabilidade de encontrar o melhor conjunto de hiperparâmetros. Em muitos casos, a busca randômica é melhor do que a busca em grade, mas ainda assim é um método computacionalmente intensivo. Ela é frequentenmente aplicada como *baseline* de OHP para medida de eficiência de novos algoritmos projetados. A busca randômica geralmente leva mais tempo e recursos computacionais do que outros métodos de busca (YU; ZHU, 2020).

Em outras palavras, a busca randômica pega amostras em um espaço de busca e avalia conjuntos de uma distribuição probabilística específica. Ou seja, a busca randômica é uma técnica em que combinações randômicas de hiperparâmetros são usadas para

encontrar a melhor solução para o modelo em consideração. O número de avaliações da busca randômica é configurada antes do processo de otimização iniciar. Dessa forma, a complexidade deste algoritmo, que é bem menor do que a busca em grade, é $O(n)$. A desvantagem é que este algoritmo não leva em consideração a informação obtida pelas avaliações do conjunto de hiperparâmetros para escolher os próximos conjuntos mais promissores na próxima rodada de avaliações (ELGELDAWI et al., 2021).

2.6.3 Otimização bayesiana

Otimização Bayesiana (OB) é um método baseado em modelos sequenciais com o objetivo de encontrar o ótimo global com o mínimo número de tentativas. OB é melhor do que a busca em grade e busca randômica por duas razões: não é necessário que o usuário possua conhecimentos preliminares da distribuição dos hiperparâmetros e também por conta da probabilidade posterior que é quando um conjunto de hiperparâmetros são escolhidos baseados nos resultados dos modelos anteriores. Essa probalidade posterior é a ideia chave da OB. Comparado com as buscas randômicas e em grade, a OB é computacionalmente menos custosa e alcança melhores conjuntos de hiperparâmetros com menos tentativas (YU; ZHU, 2020).

O que o OB faz é escolher um modelo "substituto" (*surrogate* na literatura) à função objetivo que se quer otimziar. A ideia é gastar mais tempo selecionando os próximos hiperparâmetros de forma a fazer menos chamada à função objetivo. O tempo gasto selecionando os próximos hiperparâmetros é muito pequeno comparado ao tempo gasto na função objetivo. Sem falar que é muito mais fácil otimizar a função *surrogate* do que a função objetivo (KOEHRSEN, 2018).

O *Sequential Model-Based Optimization* (SMBO) é um formalismo do OB. Esse algoritmo, basicamente, realiza cálculos para obter os melhores hiperparâmetros em cada iteração, enquanto aplica a lógica *bayesiana* e atualiza o modelo *surrogate*. Existem variantes do método do SMBO no sentido de qua modelo *surrogate* da função objetivo é utilizado e o critério para selecionar os próximos hiperparâmetros. Os exemplos mais comuns de modelos *surrogate* são *Gaussian Processes, Random Forrest Regressions e o Tree Parzen Estimator* (TPE). A escolha mais comum para o critério utilizado de seleção dos próximos hiperparâmetros é o *Expected Improvement* que é dado como a seguinte

equação:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y)p(y|x)dy \tag{2.10}$$

onde y^* é o valor limite da função objetivo, x é o conjunto proposto de hiperparâmetros, y é o valor atual da função objetivo usando os hiperparâmetros x e $p(y|x)$ é o modelo de probabilidade *surrogate* expressando a probabilidade de y dado x. Em outras palavras, o objetivo é a maximização do *Expected Improvement* em relação ao x (KOEHRSEN, 2018). A complexidade de tempo das otimizações bayesianas são $O(n^k)$ (ELGELDAWI et al., 2021).

2.6.4 Tree Parzen Estimator

O *Tree Parzen Estimator* (TPE) é um modelo com uma estrutura gráfica que lida com espaço de busca condicionais e é uma alternativa para modelos substitutos como o OB (YU; ZHU, 2020). O TPE apresenta melhores resultados do que o método Bayesiano ao lidar com variáveis condicionais por causa de sua estrutura em árvore (STROBL et al., 2008).

O TPE é um tipo de modelo *surrogate* que constrói o modelo aplicando o *Bayes rule* que, diferente da probabilidade direta $p(y|x)$ é dada como:

$$p(y|x) = \frac{p(x|y) * p(y)}{p(x)} \tag{2.11}$$

onde $p(x|y)$ é a probabilidade dos hiperparâmetros dada uma pontuação na função objetivo (KOEHRSEN, 2018). (BERGSTRA et al., 2013) utiliza o TPE e, comparando com utilizar um modelo *surrogate* como o TPE e entre realizar os cálculos utilizando a função objetivo, usar o TPE atinge melhores desempenhos chegando a finalizar o cálculo em segundos enquanto o outro método leva horas pra terminar de ser calculado.

2.6.5 Sucessive Halving e HyperBand

Os algoritmos *Sucessive Halving (SHA)* (JAMIESON; TALWALKAR, 2015) e *HyperBand* (HB) são baseados em problemas *multi-bandit*. Processos *bandit* são um tipo

especial de Processo de Decisão de Markov com várias escolhas possíveis (KAUFMANN; GARIVIER, 2017) e tem uma longa história de configuração estocástica. O algoritmo SHA foi construído sobre uma formulação *multi-armed bandit* e funciona basicamente avaliando a performance de todas as tentativas de um conjunto de configurações de hiperparâmetros, elimina a metade dos piores resultados e repete esse processo até permanecer apenas uma tentativa. Comparado ao OB, o SHA é teoricamente mais fácil e mais computacionalmente eficiente. O algoritmo HB é uma extensão do SHA. O HB tem basicamente duas melhorias em relação ao SHA: 1) pode haver várias combinações de número de tentativas e configurações; 2) o usuário pode escolher o percentual de tentativas descartadas. Comparado com a busca randômica e o OB, o HB possui uma maior acurácia com menos recursos (YU; ZHU, 2020).

O HB foi posteriormente melhorado e resultado em algoritmos considerados extensões do HB, tais como o SHA assíncrono (ASHA, na silga em inglês) e o BOHB que é uma fusão do HB com o OB. O ASHA melhora o HB aumentando a eficiência de paralelização assíncrona. Para paralelizações em larga escala, o ASHA ultrapassa os resultados estados-da-arte. O BOHB utiliza o método ETP, na parte do OB e atinge rápida convergência para os melhores conjuntos de hiperparâmetros com métodos Bayesianos, e a viabilidade de paralelização do HB (YU; ZHU, 2020).

2.6.6 Algoritmos genéticos

Algoritmos metaheurísticos são principalmente aquelas abordagens inspiradas por teorias biológicas. São conhecidos pela habilidade de resolver problemas de otimização não-contínuos e não-convexos. Algoritmos metaheurísticos iniciam-se com a criação de uma população que representa uma geração a cada iteração. Cada geração consiste de um número de candidatos de possíveis soluções chamados de indivíduos, nos quais tem um conjunto de propriedades representados pelos cromossomos. Cada indivíduo de cada geração são avaliados até que o ótimo global seja alcançado (ELGELDAWI et al., 2021).

Algoritmos genéticos (AG) é baseado na teoria evolucionária de que os indivíduos com melhores capacidades de se adaptar a um ambiente são mais aptos a sobreviver e, assim, passam suas capacidades para as próximas gerações. A cada geração são produzidos diferentes indivíduos com as capacidades dos indivíduos da geração passada. Os indivíduos

com as melhores capacidades são mais propensos a alcançar um estado de ótimo global. No AG, os hiperparâmetros são representados pelos cromossomos. Os métodos de seleção dos cromossomos, *crossover* e operações de mutação são realizados nos cromossomos para obtenção dos melhores hiperparâmetros. Os cromossomos com melhores valores são selecionados para os métodos citados acima e, consequente criação da proxima geração. O *crossover* e a mutação garantem a diversidade da próxima geração, reduzindo a chance de negligenciar boas características (YANG; SHAMI, 2020). No AG, a configuração de uma boa inicialização é um importante passo pois torna a convergência para um valor ótimo mais rápida. Uma vantagem do AG é que a inicialização acontece de forma randômica, necessitando pouco esforço ao escolher bons valores iniciais. O tempo de complexidade do AG é de $O(n^2)$ (LOBO et al., 2000) que é considerado uma velocidade de convergência baixa (ELGELDAWI et al., 2021).

2.6.7 Particle Swarm Optimization

Outro algoritmo metaheurístimo é o *Particle Swarm Optimization* (PSO) que é usado comumente para resolução de problemas de otimização. PSO resolve um problema tentando otimizar uma solução de forma iterativa com respeito a alguma medida de qualidade habilitando um grupo de partículas (*swarm*) para checar o espaço de busca de forma semi-automática (CHUAN; QUANYUAN, 2007). O algoritmo de PSO encontra uma solução otimizada por meio do compartilhamento de informações e cooperação entre partículas num grupo. Em PSO, um *swarm* tem um grupo de partículas e cada partícula é representada por um vetor contendo a posição e velocidades atuais e a melhor posição conhecida da partícula. Depois de inicializar a posição e velocidades de cada partícula, a posição atual e o desempenho de cada partícula é calculado. Na próxima iteração, a velocidade da partícula é recalculada baseada nas informações das iterações anteriores, tais como a posição e a posição ótima global corrente. Esses passos são repetidos até atingir alguma convergência ou critério de término. PSO é mais fácil de implementar do que o AG. Enquanto que no AG todos os cromossomos compatilham informações entre si, no PSO apenas as partículas com melhores desempenhos compartilham informações e as informações da melhor partícula global é transmitida às outras partículas. A complexidade computacional do PSO é de $O(nlogn)$ (YAN et al., 2017). Dessa forma, geralmente o

PSO tem uma convergência mais rápida do que o AG. Além do mais, o PSO é fácil de paralelizar desde que as partículas operam independentemente e só precisam compartilhar informações com cada um depois de cada iteração (YANG; SHAMI, 2020), ao contrário dos AG. A desvantagem do PSO é que ele precisa de uma inicialização apropriada, caso contrário, só atingirá os ótimos locais de uma função. Existem estratégias de inicialização apropriadas para o PSO (RAUF et al., 2020), mas que podem piorar sua performance (ELGELDAWI et al., 2021).

2.6.8 Estratégia de Early Stopping

Além da estratégia de algoritmos de otimização de hiperparâmetros existem estratégias de *early stopping* que é uma série de métodos que tentam simular o comportamento de especialistas de IA que diminuem o espaço de busca ao avaliar o modelo durante o treinamento e decidir se devem parar ou deixar que o modelo continue no treinamento. O objetivo disso é maximizar os recursos computacionais para os conjuntos de hiperparâmetros mais promissores. O algoritmo mais básico que utiliza a política do *early stopping* é a parada mediana. A parada mediana toma uma decisão baseada na métrica da acurácia ou *loss* reportado nas computações anteriores. Se esses valores forem menor do que a mediana da média de todas as tentativas completas, a tentativa em questão é parada (YU; ZHU, 2020). Esses algortimos também são chamados de algoritmos de poda.

2.7 Frameworks de Software

Os frameworks são de suma importância para o desenvolvimento de sistemas de software orientado a objetos em larga escala. Eles prometem uma alta produtividade e menos tempo para desenvolvimento e se beneficia do reuso de código (RIEHLE, 2000). Um framework orientado a objeto possui um *design* reusável junto com a implementação. O *design* representa um modelo de um domínio de aplicação ou um aspecto pertinente do mesmo. A implementação define como o modelo pode ser executado, pelo menos parcialmente. Um bom *design* e implementação de frameworks é o resultado de um entendimento profundo do domínio da aplicação. Esse entendimento é frequentemente

ganho no desenvolvimento de várias aplicações para aquele domínio. O framework representa a experiência acumulada de como a arquitetura de software e sua implementação para a maioria das aplicações em que um domínio deve parecer (RIEHLE, 2000).

Os desenvolvedores que utilizam um framework reutilizam seu *design* e implementação. Os desenvolvedores customizam uma arquitetura de software para o seu problema de aplicação específico. Por meio do reuso do código e *design*, os desenvolvedores adquirem alta produtividade e menor tempo de mercado (RIEHLE, 2000).

Os frameworks são similares às bibliotecas, mas existem algumas diferenças, tais como:

- Comportamento Padrão: antes da customização, um framework se comporta de uma maneira específica para a ação do usuário;
- Controle de inversão: Ao contrário de outras bibliotecas, o fluxo de controle global dentro do framework é empregado pelo framework ao invés do método que executa;
- Extensibilidade: um usuário pode extender o framework ao substituir, seletivamente, o código do framework com o próprio código;
- Código do Framework não-modificável: um usuário pode extender o framework mas não modificar o código. O propósito do framework de software é simplificar o ambiente de desenvolvimento, permitindo que desenvolvedores se dediquem aos seus esforços de requisitos de projeto, do que lidar com funções repetitivas e bibliotecas (MWENDI, 2014).

Depois de falar sobre as características do framework em relação às bibliotecas, em seguida, são listadas algumas vantagens e desvantages de se utilizar um framework:

- Vantagens
  - reuso de código que tem sido pré-processado e pré-testado. Isso melhora a confiabilidade da nova aplicação e reduz o esforço de programação e de teste;
  - um framework pode ajudar a estabelecer melhores práticas de programação e uso apropriado de padrões de *design* e novas ferramentas de programação;
  - um framework pode oferecer novas funcionalidades, melhorar o desempenho, ou melhorar a qualidade sem programação adicional pelo usuário do framework;
  - por definição, um framework oferece os meios de extender seu comportamento.
- Desvantagens
  - criar um framework é difícil e custa tempo e dinheiro;

- a curva de aprendizado para um novo framework pode ser íngreme;
- ao longo do tempo, um framework pode se tornar cada vez mais complexo (MWENDI, 2014).

2.7.1 Framework versus biblioteca

Existem muitas diferenças entre frameworks e bibliotecas que são importantes pontuar. Entre as diferenças estão:

- **Definição:** as bibliotecas fornecem aos desenvolvedores funções e classes predefinidades para ajudá-los a tornar mais fácil o desenvolvimento de aplicações. Os *frameworks* são como uma base em que os desenvolvedores construirão aplicações para plataformas específicas;
- **Inversão de controle:** com as bibliotecas é possível controlar o fluxo da aplicação e chamar a biblioteca. Já o *framework* controla o fluxo e a chamada para o código, ou seja, há uma inversão de controle;
- **Extensibilidade:** bilbliotecas, em geral, não são desenvolvidas para serem extensíveis, mas os *frameworks*, sim. *Frameworks* fornecem funcionalidades gerais para serem extensíveis;
- **Benefícios:** as bibliotecas possuem boa qualidade de código, reusabilidade e controle, melhoria da velocidade e performance dos programas, entre outras. Os *frameworks* fornecem programação mais rápida, suporte para a comunidade, grande suporte para o padrão *Model View Controller* (MVC), entre outros (ACADEMY, 2021).

2.7.2 Padrões de projeto de software

Um padrão de *design*, em engenharia de software, é uma solução geral e reusável para um problema que ocorre frequentemente no *design* do software. Um padrão de *design* é uma descrição ou um *template* de como resolver um problema que pode ser usado em várias situações (MWENDI, 2014).

Padrões de projeto foram originalmente agrupados em 3 categorias: padrões criacionais, padrões estruturais e padrões de comportamento, e descritos usando os conceitos de *delegação, agregação, e consulta.* Em padrões de projeto, uma *agregação* não é um

padrão de projeto mas se refere a um objeto, tal como uma lista ou vetor, que oferece uma interface para a criação de iteradores. *Delegação* pode ser visto como passar a execução para outro objeto enquanto retendo a identidade do objeto chamado. *Consulta* é enviar um processsamento para outro objeto que age por si só e não faz uma referência implícita ao receptor da mensagem original. Esse trabalho utiliza alguns conceitos de padrões de *design*, tais como padrões criacionais, especificamente o *Template Method* e o *Factory Method*, que é uma classe que possui um método de criação de outros objetos (MWENDI, 2014).

2.7.2.1 Template Method

O *Template Method* é um dos padrões comportamentais existentes na literatura de padrões de projeto que deixa as subclasses redefinirem certos passos de um algoritmo sem mudar a estrutura de todo algoritmo (GAMMA et al., 2021). Um exemplo de estrutura do *Template Method* se encontra na figura 12.

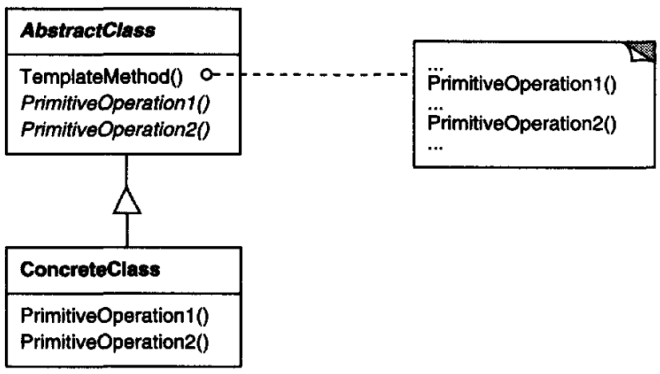

Figura 12 – Exemplo de estrutura do padrão de projeto *Template Method* (GAMMA et al., 2021).

As aplicabilidades desse padrão são: implementar as partes invariantes de um algoritmo uma vez e deixar para as subclasses os comportamentos que variam; quando comportamentos entre subclasses devem ser refatorados e colocados em uma classe comum para evitar duplicação de código; para controlar as extensões das subclasses (GAMMA et al., 2021).

Dessa forma, o *Template Method* é uma técnica fundamental para reuso de código

e são particularmente importantes em bibliotecas de classes (GAMMA et al., 2021).

2.7.2.2 Factory Method

O *Factory Method* define uma interface para criação de um objeto, mas deixa as subclasses decidirem quais classes instanciar. O *Factory Method* deixa uma classe diferir a instanciação para as subclasses (GAMMA et al., 2021). Um exemplo da estrutura do *Factory Method* se encontra na figura 13.

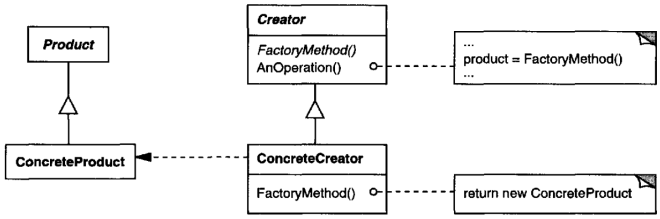

Figura 13 – Exemplo de estrutura do padrão de projeto *Factory Method* (GAMMA et al., 2021).

Usa-se o padrão de projeto *Factory Method* quando: uma classe não pode antecipar a classe de objetos que precisam ser criados; uma classe precisa que as subclasses especifiquem os objetos a serem criados por estes (GAMMA et al., 2021).

O *Factory Method* elimina a necessidade de vincular classes específicas da aplicação no código. Uma desvantagem potencial do *Factory Method* é que o cliente precisa de uma subclasse a partir da classe criadora apenas para criar um objeto concreto particular. O *Factory Method* fornece "ganchos" às subclasses provendo uma versão extendida de um objeto. O *Factory Method* também conecta hierarquias de classes paralelas. As hierarquias de classes paralelas são o resultado de uma classe que delega algumas de suas responsabilidades a outra classe separada (GAMMA et al., 2021).

3 Revisão da Literatura

Este capítulo contém a Revisão de Literatura na área de Extração de Relação, especificamente de Classificação de Relações. Primeiro são desenvolvidos os conceitos e explicações sobre arquiteturas de DL, datasets e *frameworks* utilizados para ER/CR. Este capítulo está dividido em seções que falarão sobre a revisão de literatura sobre ER/CR em relação a trabalhos relacionados usando-se modelos de DL (Seção 3.1), os *datasets* e métricas (Seção 3.2), os *frameworks* de otimização de hiperparâmetros (Seção 3.3) e, por fim, será falado sobre os *frameworks* utilizados para tarefas de CR (Seção 3.4).

3.1 Trabalhos relacionados para CR usando Deep Learning

As redes neurais trouxeram resultados bastante promissores para a área de PLN, especificamente de extração e clasificação de relações. Essas redes incluem CNNs, RNNs, redes neurais de grafos (GNNs) e *Transformers*. Nos subtópicos abaixo, serão discutidos alguns modelos de arquitetura utilizados nesse projeto, sua importância e alguns resultados.

3.1.1 CNN

A CNN processa informações estruturais e locais num texto. Dentre os modelos de CNN orientados à estrutura para tarefas de CR, o melhor deles é o CR-CNN de (SANTOS et al., 2015) com 85,6 de *F1 score* usando o *Ranking Loss* como função de *loss*, de acordo com (WANG et al., 2021). Já os modelos de CNN orientados à semântica, o melhor resultado vem do modelo *Att-Pooling-CNN* com 88 de *F1 score* usando o *Distance Function* como função de *loss* (WANG et al., 2016).

As principais realizações sobre modelos baseados em CNN foram: a criação do *position embedding* (PE) por (ZENG et al., 2014); a introdução do conceito de *multiple-window size filters* que melhora a estrutura da CNN tornando-a eficiente (NGUYEN; GRISHMAN, 2015). (SANTOS et al., 2015) melhora a função de *loss* usando outro classificador ao invés do *softmax*. Os parâmetros do modelo são treinados minimizando a função de *loss* por *ranking* (CR-CNN). Essa nova função de *loss* melhora esse modelo e pode ser usado em outros classificadores. Todos os modelos citados foram melhorados em

relação à estrutura (WANG et al., 2021).

O uso de *Shortest Dependency Path* (SDP) com CNNs foi realizado pela primeira vez por (XU et al., 2015). Esse tipo de estrutura melhora a acurácia com exemplos negativos simples. O mecanismo de atenção também melhora a representação semântica. (WANG et al., 2016) usa dois níveis do mecanismo de atenção.

O problema das CNNs é que raramente consideram *features* globais e informações de sequência de tempo especialmente para dependência entre pares de entidades. As RNNs, GRUs e LSTMs podem aliviar esses problemas (WANG et al., 2021).

3.1.2 RNN e LSTM

De acordo com (WANG et al., 2021), o melhor modelo baseado em RNN e LSTM foi o modelo orientado à semântica chamado DRNNs com 86.1 de *F1 score* de (XU et al., 2016). Apesar dos métodos usando BiLSTM melhorar a representação de *features* a nível de sentença, ainda subsiste 2 problemas: um número grande de *features* artificiais externas são introduzidas e não existe nenhum mecanismo de filtro de *features* eficaz (WANG et al., 2021).

Usa-se SDP-LSTM somado com informações externas como *word embeddings*, *POS tags*, relações gramaticais e hiperônimos do *WordNet* (XU et al., 2015). Este modelo foi melhorado com um aumento de camadas de rede neural que capturam as *features* abstratas ao longo de 2 sub-caminhos do SDP (XU et al., 2016). O SDP filtra o texto de entrada mas não filtra as *features* extraídas. (ZHOU et al., 2016) propõe o BiLSTM com mecanismo de atenção para resolver esse problema. Esse modelo automaticamente evidencia as *features* importantes apenas com o texto "puro" sem quaisquer outros *kits* de ferramenta ou recursos léxicos de PLN. Um trabalho similar é o de (XIAO; LIU, 2016) que usa uma BiLSTM com dois níveis de mecanismo de atenção para extrair uma representação de alto nível da sentença bruta (WANG et al., 2021).

(QIN et al., 2017) usa o EAtt-BiGRU que, ao contrário do BiLSTM de (ZHOU et al., 2016) utiliza *Gated Recurrent Unit* (GRU) para reduzir a computação e ajuda a adotar uma GRU de caminho para extrair conhecimento prévio dos pares de entidade. Dessa forma, esse modelo pode gerar os pesos de atenção correspondentes de forma adaptativa (WANG et al., 2021).

(ZHANG et al., 2019) propôs outro tipo de mecanismo de atenção baseado em SDP com conhecimento prévio. Esse modelo usa BiGRU para extrair *features* a nível de sentença e pesos de atenção para selecionar *features* de CNN multi-canal para classificação. Comparado com outros mecanismos de atenção aleatório ou baseado em entidade, esse modelo contrói um melhor peso de atenção usando SDP (WANG et al., 2021).

3.1.3 CNN + LSTM

(ZHENG et al., 2016) propôs 2 redes neurais baseadas em CNN e LSTM unindo o aprendizado de padrões de relações e entidade semântica. Com isso, a performance de ER pôde ser melhorada. Essa pesquisa lança uma nova luz para a fusão dos dois módulos, mostrando assim, a complementariedade dos módulos. (ZHANG; XIANG, 2018) introduz o BiLSTM-CNN, sem quaisquer mecanismos léxicos de atenção, usando apenas 3 tipos de recursos como o *word embedding*, o *position embedding*, *position indicators*. Isso mostra que uma simples combinação de ambos os módulos como LSTM e CNN pode dar uma maior performance comparado à performance de modelos mais simples.

(CAI et al., 2016) combina CNNs e RNNs dependendo do SDP. Para melhorar o conhecimento do modelo em relação a direcionalidade da relação, o modelo, chamado de BRCNN, aprende *features* do SDP em ambas as direções positivas e negativas, o que é benéfico para a predição da direção da relação. (REN et al., 2018) inspirado por esse trabalho, usa dois tipos de atenção (*intra-cross*) para combinar *features* de classificação que vem de sentenças originais e suas descrições correspondentes. (GUO et al., 2019a) propôs um novo modelo Att-RCNN para extrair *features* do texto.Esse modelo alavanca as unidades LSTM e usa CNNs mais eficientes para extrair *features* de alto nível. A parte especial desse modelo é a introdução de um novo método "contra-ruídos" que pode ser um fragmento contínuo do texto original baseado em SDP (XU et al., 2015). (WANG et al., 2020) utiliza o SDP para construir um Bi-SDP com pesos de atenção paralelo para lidar com a direção da relação. Esse método introduz mais informações sobre SDP e interpreta outra função de preposição em uma sentença, que é ignorada em trabalhos anteriores.

3.1.4 GNN

No contexto de GNNs, surgiram alguns modelos utilizando essa abordagem que tiveram bons resultados. (ZHANG et al., 2018) utilizou GCN para realizar a tarefa de ER. Essa GCN opera convoluções nos grafos de dependência (explicitados no capítulo 2) das sentenças. (ZHANG et al., 2018) acrescentou também uma camada de BiLSTM para adquirir informações contextualizadas e diminuir os erros provocados pela geração da árvore sintática. Esse modelo teve como resultados 64 e 66,4 de *F1 score* para os datasets *TACRED* (ZHANG et al., 2017) e *SemEval 2010 Task-8* (HENDRICKX et al., 2010), respectivamente.

(WU et al., 2019) simplifica o GCN reduzindo o procedimento de transformação dos vetores de *features* para um único passo, enquanto os GCNs anteriores repetem os cálculos dos vetores de *features* em K camadas para depois obter a classificação. O *Simple Graph Convolution* (SGC) proposto por (WU et al., 2019) obtém melhores resultados do que os GCNs base para vários datasets.

O mesmo autor melhorou o modelo acrescentando uma camada de atenção na arquitetura *Contextualized Attention Guided Graph Convolution Neural Networks* (C-AGGCN) (GUO et al., 2019b). A arquitetura da C-AGGCN possui uma GCN mais uma camada de atenção e uma BiLSTM para capturar representações de contexto que são subsequentemente alimentadas pelas camadas do AGGCN. Esse modelo foi testado nos *datasets TACRED* e *SemEval 2010 Task-8* e atingiu 69 e 85,7 de F1 score para os respectivos datasets. Esse modelo atingiu estado-da-arte em relação a outros modelos utilizando CNNs e LSTMs, combinados ou não, além dos modelos que utilizam GNN.

Ao contrário de (ZHANG et al., 2018), (SAHU et al., 2020) utiliza uma abordagem diferente. Ao invés de utilizar grafos de dependência gerados por ferramentas de PLN, que são computacionalmente custosas e podem gerar erros por não serem totalmente treianadas ponta-a-ponta, o *Contextualized Self-determined Graph Convolution Network* (C-SGCN), modelo proposto por (SAHU et al., 2020), dinamicamente auto-determina múltiplos grafos ponderados usando um mecanismo de auto-atenção *multi-head* (VASWANI et al., 2017) e aplica GCN em cada um separadamente. Esse modelo atinge resultados estados-da-arte para modelos CNNs e LSTMs, combinados ou não, para o *dataset TACRED* (67,8 de *F1 Score*) e o mesmo resultado para o C-AGGCN (GUO et al., 2019b) reproduzido por (SAHU et al., 2020). A vantagem desse modelo é que ele não utiliza ferramentas PLN

para geração dos grafos de dependência de sentenças, o que o torna computacionalmente menos custoso do que o C-AGGCN.

Existem outros trabalhos (ZHU et al., 2019; BASTOS et al., 2020) que utilizam a arquitetura do GNN para tarefas de ER para outros datasets como o *Wikidata* (VRANDECIC, 2012) e o *Freebase* (BOLLACKER et al., 2007).

3.1.5 Transformers

O BERT é também bastante utilizado em ER. Um dos trabalhos que utiliza o BERT para CR é o de (SOARES et al., 2019). O modelo de (SOARES et al., 2019) ultrapassa em performance os modelos anteriores ao dele nos *datasets SemEval 2010 Task 8* (HENDRICKX et al., 2010), *KBP37* (ZHANG; WANG, 2015) e o *TACRED* (ZHANG et al., 2017). O modelo de (SOARES et al., 2019) utiliza o BERT com *tokens* de representações de marcadores de entidade e obtém como saída a representação da entidade inicial da sentença. Esse tipo de entrada e saída apresenta melhores resultados. Esse modelo é utilizado no presente trabalho para obtenção dos melhores resultados do *framework*. Existem muitos outros trabalhos que utilizam o BERT para tarefas de ER usados para o *dataset SemEval 2010 Task 8* (ZHAO et al., 2021; LI; TIAN, 2020; TAO et al., 2019) e para o *dataset DDI Extraction 2013* (ASADA et al., 2020; BOUKKOURI et al., 2020).

3.2 Datasets e métricas

Foram criados alguns *datasets* para as tarefas de ER, especificamente de CR. Esses *datasets* foram liberados para competições entre pesquisadores de PLN de diferentes partes do mundo. Os *datasets* utilizados no presente trabalho foram: *SemEval 2010 Task-8*, *SemEval 2018 Task-7* e o *DDI Extraction 2013*. As competições visavam obter os melhores modelos com os melhores resultados de métricas como *micro-F1* (para os datasets *SemEval 2010 Task-8* e *DDI Extraction 2013*) e *macro-F1* (para o dataset *SemEval 2018 Task-7*).

3.2.1 SemEval 2010 Task-8

O *SemEval 2010 Task-8* foca nas relações semânticas entre pares de nominais. O objetivo desta competição é criar um banco de ensaios para classificação automática de relações semânticas. O dataset *SemEval 2010 Task-8* possui 9 tipos de relações semânticas de interesse geral e prático (*Cause-Effect, Component-Whole, Entity-Destination, Entity-Origin, Product-Producer, Member-Collection, Message-Topic, Content-Container* e *Instrument-Agency*) e cada instância ou sentença contém apenas 2 entidades nas quais a relação será extraída. Porém, a classificação depende da direcionalidade. Por exemplo, entre 2 entidades analisadas numa sentença, uma relação de *Cause-Effect* pode ser de uma entidade localizada antes de outra entidade ou o contrário. Dessa forma, o sistema de classificação automática teria que classificar, além do tipo da relação, a direção da relação entre as entidades. Assim, o sistema precisa classificar um total de 18 relações (2 para cada tipo de relação por conta da direcionalidade) mais o tipo *Other* quando nenhuma relação existente corresponde (HENDRICKX et al., 2010).

Existem, no total, 10.717 instâncias, sendo 8.000 instâncias de treino e 2.717 instâncias testes (HENDRICKX et al., 2010). As estatísticas das relações do *SemEval 2010 Task-8* pode ser vista na tabela 4.

A melhor relação para ser extraída é a relação de *Cause-Effect* e *Member-Collection*. A mais difícil é geralmente é a relação de *Instrument-Agency* e *Product-Producer*, apesar desta apresentar uma alta aceitação entre os anotadores (HENDRICKX et al., 2010).

Foram obtidas 152 instâncias classificadas incorretamente por todos os sistemas da tarefa do *SemEval 2010 Task-8*. Isso mostrou os limites das abordagens de modelagem por requererem mais conhecimento léxico e raciocínio complexo (HENDRICKX et al., 2010).

3.2.2 SemEval 2018 Task-7

O *dataset SemEval 2018 Task-7* foca nas análises de relações semânticas de *corpus* científicos. Ao contrário de outros *datasets* de *SemEval*, como o *SemEval 2017 Task-10* que também lida com *corpus* científico, o *SemEval 2018 Task-7* tem um conjunto de relações semânticas maior e é formado por uma coleção de resumos (*abstracts*) de artigos científicos. Contém 6 categorias discretas de relações semânticas num texto (*Usage, Topic, Model-Feature, Part-Whole, Compare e Result*) específicas de domínio científico. Este

Relação	**Frequência de treino**	**Frequência de teste**	**Total por relação**
Cause-Effect	1003/12.5%	328/12.1%	1331/12.4%
Component-Whole	941/11.8%	312/11.5%	1253/11.7%
Entity-Destination	845/10.6%	292/10.7%	1137/10.6%
Entity-Origin	716/8.9%	258/9.5%	974/9.1%
Product-Producer	717/9.0%	231/8.5%	948/8.8%
Member-Collection	690/8.6%	233/8.6%	923/8.6%
Message-Topic	634/7.9%	261/9.6%	895/8.4%
Content-Container	540/6.8%	192/7.1%	732/6.8%
Instrument-Agency	504/6.3%	156/5.7%	660/6.2%
Other	1410/17.6%	454/16.7%	1864/17.4%
Total	8000	2717	10717

Tabela 4 – Estatísticas de anotações do SemEval 2010 Task-8.

dataset não possui relação que seja inexistente, tais como *Other* no *SemEval 2010 Task-8* ou *None*. A tarefa proposta na competição foi dividida em 3 subtarefas. Isso foi feito para o fornecimento de um framework para avaliação sistemática dos passos que são necessários para uma EI completa a partir de textos científicos que são as tarefas de ER e CR. Duas dessas subtarefas (1.1 e 1.2) são focadas apenas em CR. Cada subtarefa contém 350 resumos de artigos científicos com suas respectivas instâncias e categorias de relações (GÁBOR et al., 2018). Utiliza-se apenas as subtarefas 1.1 e 1.2 no presente trabalho por que ambas focam na tarefa de CR que é o objetivo deste trabalho.

As diferenças entre as subtarefas 1.1 (ST1) e a 1.2 (ST2) é que a ST1 possui entidades manualmente anotadas em ambos os dados de treino e teste. Nos dados de treino, as relações semânticas também foram manualmente anotadas na ST1. Ao contrário da ST2 em que as entidades são automaticamente anotadas em ambos os dados de treino e teste. Nos dados de treino, as relações semânticas são manualmente anotadas entre as enitdades. Nos dados de teste de ambas as subtarefas, só são informadas as entidades anotadas mas sem os dados das relações semânticas entre as entidades (GÁBOR et al., 2018).

Na tabela 5, são apresentados a distribuição estatística das relações no *dataset SemEval 2018 Task-7*.

Relação	**Frequência de treino**	**Frequência de teste**	**Total por relação**
Sub-task 1.1			
Usage	483/39.3%	175/49.3%	658/41.6%
Topic	18/1.5%	3/0.9%	21/1.3%
Model-Feature	326/26.5%	66/18.5%	392/24.8%
Part-Whole	234/19.1%	70/19.7%	304/19.2%
Compare	95/7.7%	21/6.0%	116/7.3%
Result	72/5.9%	20/5.6%	92/5.8%
Total	1228	355	1583
Sub-task 1.2			
Usage	470/37.7%	123/34.6%	593/37.0%
Topic	243/19.5%	69/19.4%	312/19.5%
Model-Feature	175/14.0%	75/21.1%	250/15.6%
Part-Whole	196/15.8%	56/15.8%	252/15.7%
Compare	41/3.2%	3/0.9%	44/2.7%
Result	123/9.8%	29/8.2%	152/9.5%
Total	1248	355	1603

Tabela 5 – Distribuição de relações anotadas no SemEval 2018 Task-7

3.2.3 DDI Extraction 2013

O *DDI Extraction 2013* é um *corpus* anotado com 792 textos selecionados da base de dados *DrugBank* e 223 *abstracts* do *MedLine*. O *dataset* possui um total de 18.502 substâncias farmacológicas e 5.028 interações entre drogas (*DDIs*), incluindo interações farmacocinéticas (*PK*) e farmacodinâmicas (*PD*). Existem 4 tipos de entidades propostas pelo dataset *DDI 2013* para anotar as substâncias farmacológicas: droga, marca, grupo e

droga_n (drogas não aprovadas para uso humano). São também 4 os tipos de relações de interação entre drogas propostos pelo *DDI 2013*: *mechanism*, *effect*, *advice* e *int* (não tem informação adicional). O tipo *int* apresenta uma maior concordância entre os anotadores (IAA) em ambos os *datasets* (*DrugBank* e *MedLine*). O tipo de relação que apresenta a segunda maior concordância entre os anotadores é a de *advice*. O *corpus DDI-DrugBank* contém 6.795 sentenças e o *DDI-MedLine* é feito de 2.147 sentenças. O IAA, tanto para entidades e relacionamentos, do *corpus DDI-DrugBank* foi maior do que para o *corpus DDI-MedLine*. Isso se deve por conta de que a complexidade dos textos dos resumos do *MedLine* são muito maiores do que o DrugBank (HERRERO-ZAZO et al., 2013). A tabela 6 mostra o número de relações anotadas em cada *corpus* do *DDI Extraction 2013*.

Relação	**Frequência de treino**	**Frequência de teste**	**Total por relação**
Effect	1,687/41.9%	360/36.8%	2,047/41.0%
Mechanism	1,319/32.8%	302/30.9%	1,621/32.4%
Advice	826/20.6%	221/22.5%	1,047/20.9%
Int	188/4.7%	96/9.8%	284/5.7%
Total	4,020	979	4,999

Tabela 6 – Distribuição de relações anotadas no *corpus* DDI 2013.

3.2.4 Métricas

Cada *dataset* apresenta uma métrica usada para avaliação dos sistemas de *deep learning*. As métricas mais comumente usadas em sistemas de *deep learning* são: acurácia, precisão, *recall* e *F1-score* (micro e macro). As métricas que foram usadas neste trabalho foram: *micro-F1* e *macro-F1*. As equações abaixo mostram essas métricas são formalmente descritas:

$$Acurácia = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$Precisão = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F1\ score = 2 * \frac{Precisão * Recall}{Precisão + Recall} \tag{3.4}$$

$$Micro\text{-}F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.5}$$

$$Macro\text{-}F1 = \frac{1}{C} \sum_{n=1}^{C} F1_n \tag{3.6}$$

onde TP, TN, FP e FN são os verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, respectivamente. O C é o número de classes que o dataset possui. O $F1_n$ na equação 3.6 é o valor do *F1 score* de uma determinada classe. Como se pode ver, a equação do *macro-F1* (3.6) é uma simples média aritmética de todos os *F1 scores* de cada classe.

A Precisão é definida como o total de exemplos classificados corretamente para uma determinada classe dividido pelo total de exemplos que foram classificados pela mesma classe, independente de estar correto ou não. Já o *Recall* é o total de exemplos classificados corretamente dividido pelo total de exemplos da mesma classe existentes no conjunto de dados. O *F1 score* é a média harmônica entre a precisão e o *recall*.

Em geral, quando se está utilizando um *dataset* não-balanceado, ou seja, em que existem mais exemplos de uma classe que de outra, e onde todas as classes são igualmente importantes, usar o *Macro-F1* pode ser uma boa escolha. Suponha que se tem um *dataset* balanceado e se quer uma métrica que resulte numa performance geral, o *Micro-F1* pode ser uma boa escolha para esse caso. No presente trabalho, utiliza-se essas duas métricas para efeitos de comparação, além da métrica de acurácia (LEUNG, 2022).

Os datasets *SemEval 2010 Task-8* e *DDI Extraction 2013* foram avaliados com a métrica do *Micro-F1* em suas competições. Já o dataset *SemEval 2018 Task-7* foi avaliado

na competição com a métrica do *Macro-F1*. Dessa forma, utilizaremos essas duas métricas para *benchmarking* dos modelos estados-da-arte nas três competições analisadas neste trabalho, pois foram divulgados os desempenhos dos modelos das arquiteturas de modelo de DL para CR nas três competições utilizando-se as métricas *Micro-F1* (para os datasets *SemEval 2010 Task-8* e *DDI Extraction 2013*) e *Macro-F1* (para o dataset *SemEval 2018 Task-7*). Ou seja, em alguns trabalhos não se tem valores em outras métricas para comparação.

3.3 Frameworks de Otimização de Hiperparâmetros

Existem diversos *frameworks open-source* que facilitam o uso de algoritmos de OHP, tais como: *HyperOpt* (BERGSTRA et al., 2013), *Vizier* (GOLOVIN et al., 2017), *Tune* (LIAW et al., 2018), *Optuna* (AKIBA et al., 2019), entre outros. No presente trabalho, utilizou-se o *Optuna*. O *Optuna* tem algumas vantagens em relação a outros *frameworks* de OHP. Algumas dessas vantagens são: programação que permite que o usuário construa dinamicamente o espaço de busca; eficientes algoritmos de poda e de amostragem que permite customização do usuário; facilidade de configuração, arquitetura versátil que pode ser implantada para tarefas de vários tipos variando desde experimentos mais leves e simples até experimentos mais robustos que exigem computação distribuída (AKIBA et al., 2019). A tabela 7 mostra as diferentes características dos *frameworks* de otimização de hiperparâmetros.

As APIs com programação *define-by-run* permite que o usuário defina dinamicamente o espaço de busca de otimização de hiperparâmetros, ao contrário da programação *define-and-run* em que é preciso definir com antecedência o espaço de busca para depois haver a execução do programa. O *Optuna* é o único dos frameworks acima listados com o estilo de API *define-by-run*. Além disso, é possível utilizar, no *Optuna*, algoritmos de poda que param a tentativa quando veem que os resultados não são promissores, agilizando, assim, a busca de hiperparâmetros otimizados. O *framework Optuna* é *light-weight*, ou seja, necessita de poucos cálculos para executar as otimizações. No *Optuna* é possível também realizar tentativas de otimizações de forma distribuída utilizando-se de um banco de dados. Por fim, o *Optuna* fornece um painel de controle *web* para visualização e análises das otimizações em tempo real (AKIBA et al., 2019). Nem

todos os *frameworks* listados acima possuem todas as funcionalidades que o *Optuna* possui como se pode ver na tabela 7.

Framework	**Estilo da API**	**Poda**	**Light-weight**	**Distri-buído**	**Painel de controle**
SMAC (HUTTER et al., 2011)	define-and-run	Não	Sim	Não	Não
Spearmint (SNOEK et al., 2012)	define-and-run	Não	Sim	Não	Não
HyperOpt (BERGSTRA et al., 2015)	define-and-run	Não	Sim	Sim	Não
AutoTune (KOCH et al., 2018)	define-and-run	Sim	Não	Sim	Sim
Vizier (GOLOVIN et al., 2017)	define-and-run	Sim	Não	Sim	Sim
Tune (LIAW et al., 2018)	define-and-run	Sim	Não	Sim	Sim
Optuna (AKIBA et al., 2019)	define-by-run	Sim	Sim	Sim	Sim

Tabela 7 – Comparações de frameworks de otimização de hiperparâmetros (AKIBA et al., 2019)

3.4 Frameworks de ER

Nesta seção, primeiro apresenta-se vários *frameworks* de ER para desenvolver sistemas ER usando técnicas estatísticas. Depois, apresenta-se dois *frameworks* ER que permitem desenvolver sistemas ER baseado em modelos de DL: *OpenNRE* (HAN et al., 2019) e REflex (CHAUHAN et al., 2019). Esses dois últimos frameworks são apresentados em detalhes com suas forças e fraquezas. Suas funcionalidades são comparadas, e novas são definidas e incluídas no novo *framework* de ER desenvolvido neste trabalho chamado *DeepREF*.

3.4.1 Frameworks de ER orientados à estatística

O *framework* proposto por (MUZAFFAR et al., 2015) tinha como objetivo *corpus* de domínio médico e usava abordagens de *machine learning* para ER. Suas principais contribuições foram a geração de um conjunto de *features* classificado pelo *Unified Medical Language System (UMLS)*. O UMLS é um conjunto de arquivos que contêm teminologias biomédicas e de saúde (BODENREIDER, 2004) e é útil para extrair conceitos, relacionamento ou conhecimento. Eles também usam uma abordagem híbrida usando *machine learning* com *bag of words*, processamento de linguagem natural e representação semântica para extrair relações biomédicas (MUZAFFAR et al., 2015).

O *framework ENRE* foi projetado para extrair entidades e relações entre si a partir de linguagem de programação (JIN et al., 2019). Esse *framework* tem a vantagem de ser facilmente extensível para novas linguagens de programação de diferentes paradigmas. Isso é importante porque, hoje em dia, os sistemas são construídos empregando várias linguagens de programação, com diferentes paradigmas (JIN et al., 2019).

Outro *framework* usa Aprendizagem por Reforço (AR) para extrair relações. Esse *framework* trabalhou de forma hierárquica, decompondo as tarefas em nível alto e baixo para detectar relações entre duas entidades e classificar a relação, respectivamente. O uso de AR supera nos resultados em alguns modelos *baseline* até mesmo na extração de relações sobrepostas (TAKANOBU et al., 2019).

O *Relation Extraction Learning Framework (REEL)* lida com os desafios de extrair novas relações sobre novas coleções de texto definidas pelo usuário. Ele usa ferramentas de *machine learning* para processamento de texto. Algumas vantagens desse *framework* é que ele é disponível publicamente, está apto para lidar com vários tipos de formatos de texto de entrada e tem bastante modularização para facilmente usar ferramentas de processamento (BARRIO et al., 2014).

3.4.2 Frameworks de ER orientados a Deep Learning

Por conta do sucesso do uso de DL em PLN, alguns *frameworks* de ER orientados à *deep learning* tem sido propostos recentemente. Existem apenas dois *frameworks* de ER orientados à DL, para o melhor do conhecimento deste trabalho: *OpenNRE* (HAN et al., 2019) e REflex (CHAUHAN et al., 2019), os quais serão apresentados nessa seção.

3.4.2.1 OpenNRE

O *OpenNRE* é um *kit* de ferramentas extensível que permite a implementação e rápida validação de modelos de redes neurais para ER em um único *framework*. Além disso, o *OpenNRE* fornece vários módulos de funções próprias de ER e CR baseadas em *frameworks* como *Tensorflow* e *PyTorch*, utilizados para implementação de modelos neurais, mantendo uma suficiente modularidade e extensibilidade permitindo a facilidade de incorporação de novos modelos nesse mesmo *framework*. Esse *framework*, apesar de possuir outros módulos voltados para diferentes tarefas relacionadas a ER tais como ER a nível de sentença, de documento e *bag-level*, ele foi desenvolvido especialmente para ER à nível de sentença. Esse *kit* de ferramentas prioriza a eficiência operacional baseada em *PyTorch* e *Tensorflow* e permite rápidas validações e treinamentos de modelos neurais de ER (HAN et al., 2019). A figura 14 mostra a arquitetura funcional do *OpenNRE*.

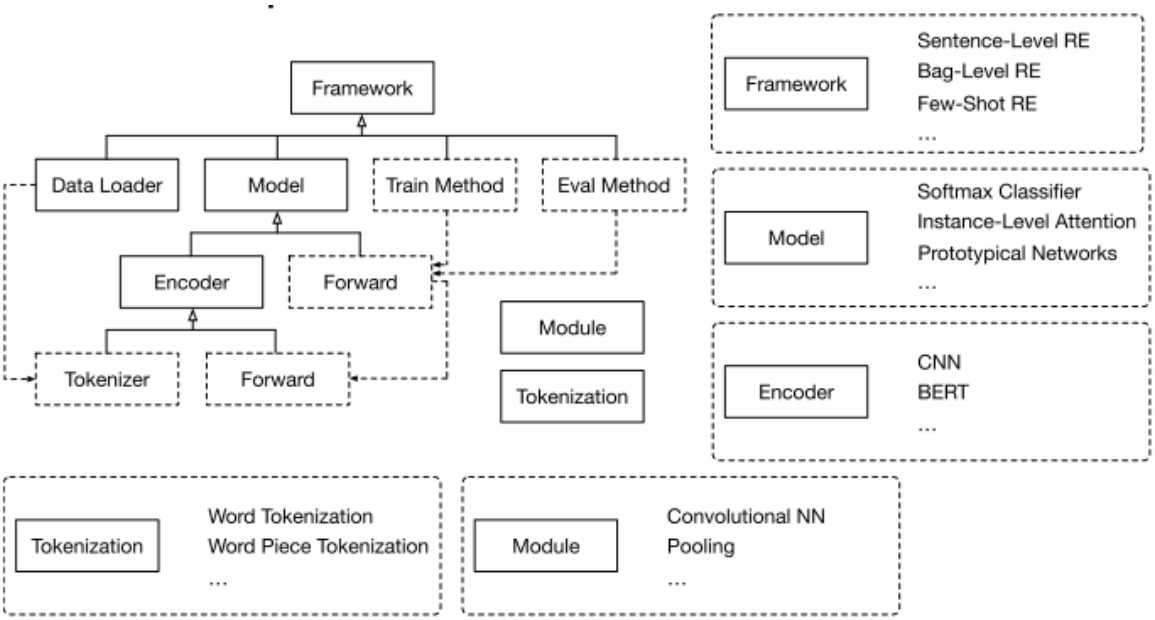

Figura 14 – Arquitetura funcional do OpenNRE (HAN et al., 2019).

No presente trabalho, houve foco na tarefa de ER a nível de sentença. No *OpenNRE* existem implementados 2 diferentes *encoders* para ER a nível de sentença: o *encoder* de CNN por (ZENG et al., 2014) e o BERT por (DEVLIN et al., 2019). As configurações do CNN para o *OpenNRE* foram extraídas de (NGUYEN; GRISHMAN, 2015), incluindo os *embeddings* de palavras e de posição. Para o BERT, foi seguido a configuração de (SOARES et al., 2019). Existem 2 tipos de encoder para o BERT no *framework* do *OpenNRE*: o

primeiro encoder, chamado apenas de BERT no artigo, usa os marcadores de entidade como entrada e obtém o *[CLS]* como saída. Existe outro encoder, chamado *BERT-Entity* (BERT-EM) no *framework*, que se refere a usar os marcadores de entidade como entrada e recebendo o início da entidade como saída do modelo como em (SOARES et al., 2019).

No framework já são pré-instalados 2 *datasets*: *SemEval 2010 Task-8* (HENDRICKX et al., 2010) e o *Wiki80*. O *Wiki80* é derivado do *FewRel* (HAN et al., 2018), o que não é o foco do presente trabalho.

3.4.2.2 REflex

O *framework REflex* foi desenvolvido com objetivos semelhantes ao presente trabalho. Esse framework objetiva identificar as fontes de variabilidade nos resultados para três *datasets* (*SemEval 2010, DDI Extraction 2013 e i2b2/VA 2010*) e fornecer um modelo base para comparação de melhorias futuras. Os objetivos de projeto desse *framework* é a identificação de melhores práticas para extração de relação e ser um guia para abordagem de novos *datasets* (CHAUHAN et al., 2019).

REflex é também um framework *open-source* e extensível para desenvolvimento de modelos baseado em DL voltados a CR (CHAUHAN et al., 2019). Este *framework* fornece um módulo que pré-processa os dados de diferentes datasets, convertendo-os em um formato de representação específica. Como resultado, é possível aplicar diferentes tipos de processamento nos dados de entrada a serem avaliados. REflex fornece módulos para lidar com entradas de dados diversos, e avaliações em modelos CR incluindo otimização de hiperparâmetros, validação cruzada, e funções de teste de significância estatística. Esses módulos permitem *ablation studies*, e comparação de desempenho direto de modelos ER. Além disso, reprodutibilidade e extensibilidade são possíveis utilizando o framework REflex. REflex fornece um módulo para usar, treinar e avaliar modelos. Contudo, não fornece encapsulação de sistema, e extensibilidade de modelos como o OpenNRE (CHAUHAN et al., 2019).

A conclusão da pesquisa de (CHAUHAN et al., 2019) foi que: 1) escolhas de pré-processamento podem causar grandes variações no desempenho dos modelos; 2) reportar os resultados em uma divisão de conjunto de teste é problemático devido ao viés causado pela divisão das sentenças no *dataset*; 3) *embeddings* contextualizados são geralmente úteis,

mas a técnica de utilizar os *embeddings* pré-treinados para treinamento é importante 4) selecionar os hiperparâmetros corretos para um dataset é importante para o desempenho 5) selecionar as métricas de avaliação corretas para um novo dataset devem levar em consideração classes não-balanceadas para classes escolhidas a serem avaliadas (CHAUHAN et al., 2019). Após várias combinações de diferentes abordagens, foram obtidos resultados estados-da-arte para os três *datasets* usados: *SemEval 2010 task 8* (HENDRICKX et al., 2010), que é um *dataset* de domínio geral, atingiu 85,87% de macro-F1 e 86,69% de micro-F1; o *dataset DDI Extration 2013* (HERRERO-ZAZO et al., 2013), de domínio biomédico, atingiu 86,53% de macro F1 e 92,72% de micro-F1; o *dataset i2b2/VA 2010* (UZUNER et al., 2011), de domínio clínico, chegou a 84,46% de macro F1 e 86,26% de micro-F1 (CHAUHAN et al., 2019).

A tabela 8 não apenas apresenta a comparação entre funcionalidades do *OpenNRE* e *REflex*, mas também as funcionalidades disponíveis no *DeepREF*. A facilidade de uso e configuração será explanada no capítulo 4 na seção sobre o Estudo de Caso Explicativo (Seção 4.6).

	OpenNRE	***REflex***	***DeepREF***
Ferramenta PLN	Nenhum	SpaCy	SpaCy, **Stanza**
Métricas de avaliação	Micro-F1	Micro e Macro-F1; Matriz de confusão	Micro, Macro e **Weighted**-F1; Matriz de confusão
Domínio	Geral	Geral, Biomédico e Clínico	Geral, Biomédico e **Científico**
Modelo de aprendizagem	CNN, PCNN e BERT	CRCNN	CNN, PCNN, CRCNN, BERT, **GRU/BiGRU**, **LSTM/BiLSTM**
Embedding	Glove e BERTbase	Senna, ELMo, BERT-Tokens	Glove, Senna, BERTbase, **FastText Wiki, FastText Crawl, BioBERT, SciBERT, Sentence-BERT, Semântico (WordNet), POS Tag, Grafos de dependência**
Processamento de texto	Nenhum	Ofuscamento de Entidades, NER e Dígitos; Remoção de pontuação e *stopwords*	Ofuscamento de Entidades, NER e Dígitos; remoção de **texto entre parênteses ou colchetes**, de pontuação e de *stopwords*
Modularizado	Sim	Não	Sim
Configuração	Fácil	Difícil	Fácil
Otimizável	Não	Sim (sem algoritmo de poda)	Sim **(com algoritmo de poda)**
Processamento linguístico	tokenização	tokenização, NER, *POS tag*	tokenização, NER, *POS tag*, **Parsing de dependência, WordNet**
Datasets de avaliação	SemEval 2010 Task-8, Wiki80, TACRED	SemEval 2010 Task-8, DDI Extraction 2013, i2b2	SemEval 2010 Task-8, **SemEval 2018 Task-7**, DDI Extraction 2013

Tabela 8 – Comparação de frameworks de CR baseados em DLs (NASCIMENTO et al., 2022).

4 DeepREF - Um Framework para Classificação de Relações

Neste capítulo será explanado a metodologia de construção do novo *framework*, o *DeepREF*, que é uma melhoria do *framework OpenNRE* (HAN et al., 2019) com alguns módulos do *REflex* (CHAUHAN et al., 2019). O código do *framework DeepREF* tem como base o código do *OpenNRE*, herdando, assim, encapsulamento, extensibilidade e facilidade de uso deste. A arquitetura funcional do *DeepREF* está retratada na figura 16.

A seção 4.1 aborda sobre o *framework* em si comparando com os outros tipos de *frameworks*. Nas demais seções é falado sobre os módulos existentes no *DeepREF*.

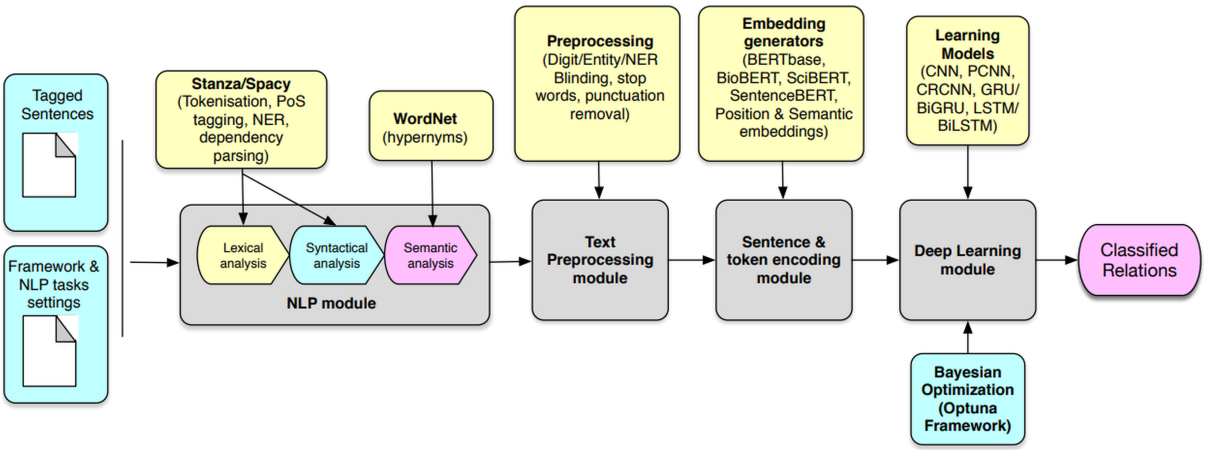

Figura 15 – Arquitetura funcional do *DeepREF* (NASCIMENTO et al., 2022)

4.1 Comparações entre frameworks

Nesta seção compara-se as forças e fraquezas do *OpenNRE* e o *REflex*, e compara-se também suas funcionalidades com as funcionalidades do presente *framework DeepREF*.

O *framework OpenNRE* tem três vantagens comparado ao *framework REflex*. Primeiro, o *OpenNRE* implementa vários modelos de ER estados-da-arte, incluindo mecanismo de atenção, *adversarial learning*, e aprendizagem por reforço. Segundo, *OpenNRE* aprecia de um ótimo encapsulamento de sistema. Ele divide o *pipeline* da extração de relação em quatro partes, a saber, *embedding*, *encoder*, *selector* (para *distant supervision*), e *classifier*. Para cada parte, possui vários métodos

implementados. Encapsulamento de sistemas torna fácil treinar e testar modelos alternando os hiperparâmetros ou apontando arquiteturas de modelo usando argumentos em *Python*. Terceiro, *OpenNRE* é extensível. Usuários podem construir novos modelos de ER escolhendo blocos específicos disponíveis em quatro partes como mencionado acima e combiná-los livremente, com apenas algumas linhas de código. Apesar de muitas vantagens, o OpenNRE não possui módulos que tratem dos dados dos datasets para pré-processamentos e avaliações entre eles para fins de comparação.

Por outro lado, o *framework REflex*, ao contrário do *OpenNRE*, permite reproduzir diferentes experimentos a partir de diferentes modelos e *datasets*. Além do mais, *REflex* tem um módulo de pré-processamento eficiente para alguns *datasets* públicos e é possível extender para outros *datasets*. Apesar de possuir diferentes tipos de pré-processamento (ofuscamento de dígitos, remoção de pontuação e *stopwords*, e substituição de ofuscamento de entidades ou Reconhecimento de Entidades Nomeadas - NER, no inglês), existem outras formas de comparar e melhorar os resultados do modelo como ajustes de hiperparâmetros, divisão de viés de conjunto de treino e teste para checar a significância estatística e usos diferentes de *word embeddings* tais como *ELMo* e *BERT-tokens* (CHAUHAN et al., 2019). O número de *embeddings* disponível no *REflex* é mais importante do que no *OpenNRE*. Apesar do REflex ter como objetivo os comparativos de diferentes datasets e ser extensível, ele não possui bom encapsulamento. Os métodos para pré-processamento não são organizados em classes e objetos, mas foram desenvolvidos para serem uilizados de forma sequencial.

A implementação das arquiteturas dos modelos e do processo de treinamento é principalmente baseados no *OpenNRE*. A figura 16 mostra a arquitetura de classes do *DeepREF* o que foi acrescentado tendo como base a arquitetura do *OpenNRE*.

O que está escrito em vermelho na figura 16 é o que foi acrescentado no framework. Os novos módulos acrescentados foram os módulos de *Dataset*, *Optimization* e *NLP*. Esses 3 módulos estão relacionados com os módulos PLN, de pré-processamento e o módulo de *Deep Learning*, respectivamente, que serão explanados nas seções seguintes. O módulo de *encoding* de sentença e *token* está atrelado ao módulo de *Deep Learning*.

Os módulos dos convertedores e pré-processamento foram inspirados nos módulos do *REflex*.

A arquitetura *DeepREF* é composta de quatro principais módulos: “módulo PLN”,

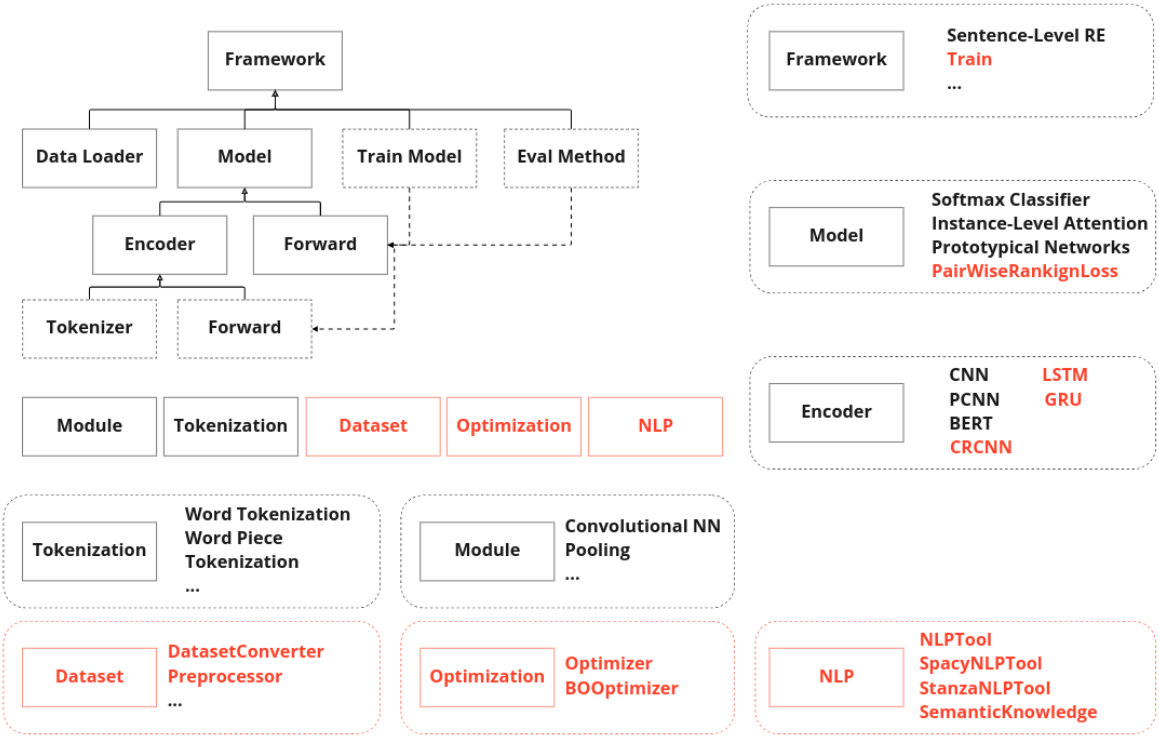

Figura 16 – Arquitetura de classes do *DeepREF*

"módulo de pré-processamento", "módulo de encoding de sentença e *token*" e o "módulo Deep Learning" (NASCIMENTO et al., 2022). Eles são descritos nas próximas seções.

4.2 Módulo PLN

O módulo PLN é responsável por pegar sentenças do dataset e obter informações, tais como *tokenização*, com *part-of-speech* (POS) *tags*, rótulos de dependência, e entidades nomeadas (NER). Ou seja, nesse módulo são realizadas análises léxicas e sintáticas os quais anotam as sentenças obtidas pelos arquivos de datasets brutos e transformam num arquivo de dataset padrão com todas as informações do dataset como as sentenças *tokenizadas*, as entidades das sentenças que estão sendo analisadas para classificação de relações, a posição e nome das sentenças, o nome da relação entre as entidades na sentença, os *part-of-speech tags* de cada *token* na sentença, os rótulos de dependência de cada *token* na sentença, os rótulos do reconhecimento das entidades nomeadas (NER, na sigla em inglês) e, por fim, os hiperônimos de cada entidade-candidato na sentença até 2 níveis acima, respectivamente.

Os hiperônimos são obtidos por uma análise semântica utilizando-se o *WordNet* da

biblioteca do NLTK[1].

Apesar do *Stanza* ser disponível no *DeepREF*, o *SpaCy* foi escolhido para realizar as tarefas de PLN por ser mais rápido para realizar as análises léxicas e sintáticas do que o *Stanza* As classes no *DeepREF* responsáveis por converter o *dataset* bruto para um formato padrão são as classes dos *converters*. Cada *dataset* tem a sua própria classe *converter*. Essas classes herdam os métodos e atributos da classe *DatasetConverter* e estão dispostas segundo um padrão de projeto chamado *factory method*, que é um padrão de projeto do tipo criacional, em que são criadas as classes que representam cada *dataset* disponível no *framework*. Cada classe de *dataset* herda da classe *Dataset* com seus métodos e atributos. Cada classe *Dataset* possui uma lista de classes *Sentence* que é responsável por definir as informações básicas da sentença como *tokens*, nome e posição das entidades das quais será extraída a relação entre elas, o nome do tipo de relação e informações de relações sintáticas e semânticas das entidades, tais como os POS *tags*, os rótulos de dependências, a categorização da sentença e as informações semânticas das entidades que são os 2 hiperônimos no primeiro nível (pai) e segundo nível (avô). Por exemplo, se existe uma entidade chamada "carro", um possível hiperônimo pai seria uma palavra mais geral que significasse a palavra da entidade que seria "automóvel", por exemplo. E o hiperônimo avô seria "máquina", por exemplo, pois possui um significado mais geral do que a palavra "automóvel". A figura 17 mostra o diagrama de classes do módulo PLN.

4.3 Módulo de processamento de texto

Esse módulo é inspirado no *REflex* com melhorias em relação à organização e distribuição de tarefas nas suas respectivas classes. As tarefas de pré-processamento seguintes estão disponíveis no *REflex* e que também foram integradas no nosso trabalho: *punct_digit*[2], *puct_stop_digit*[3], *entity_blinding*[4].

De fato, foi refatorado o código *REflex* para tornar possível uma combinação de todas as tarefas de pré-processamento acima mencionadas. Por exemplo, é possível no *DeepREF* combinar remoções de pontuação, *stopwords* e ofuscamento de dígitos e entidades, usando ou não o NER, desde que essas duas tarefas são mutualmente exclusivas porque

[1] <https://www.nltk.org/howto/wordnet.html>
[2] ofuscamento de dígitos e remoção de pontuação
[3] ofuscamento de dígitos e remoção de *stopwords* e pontuação
[4] ofuscamento de entidades

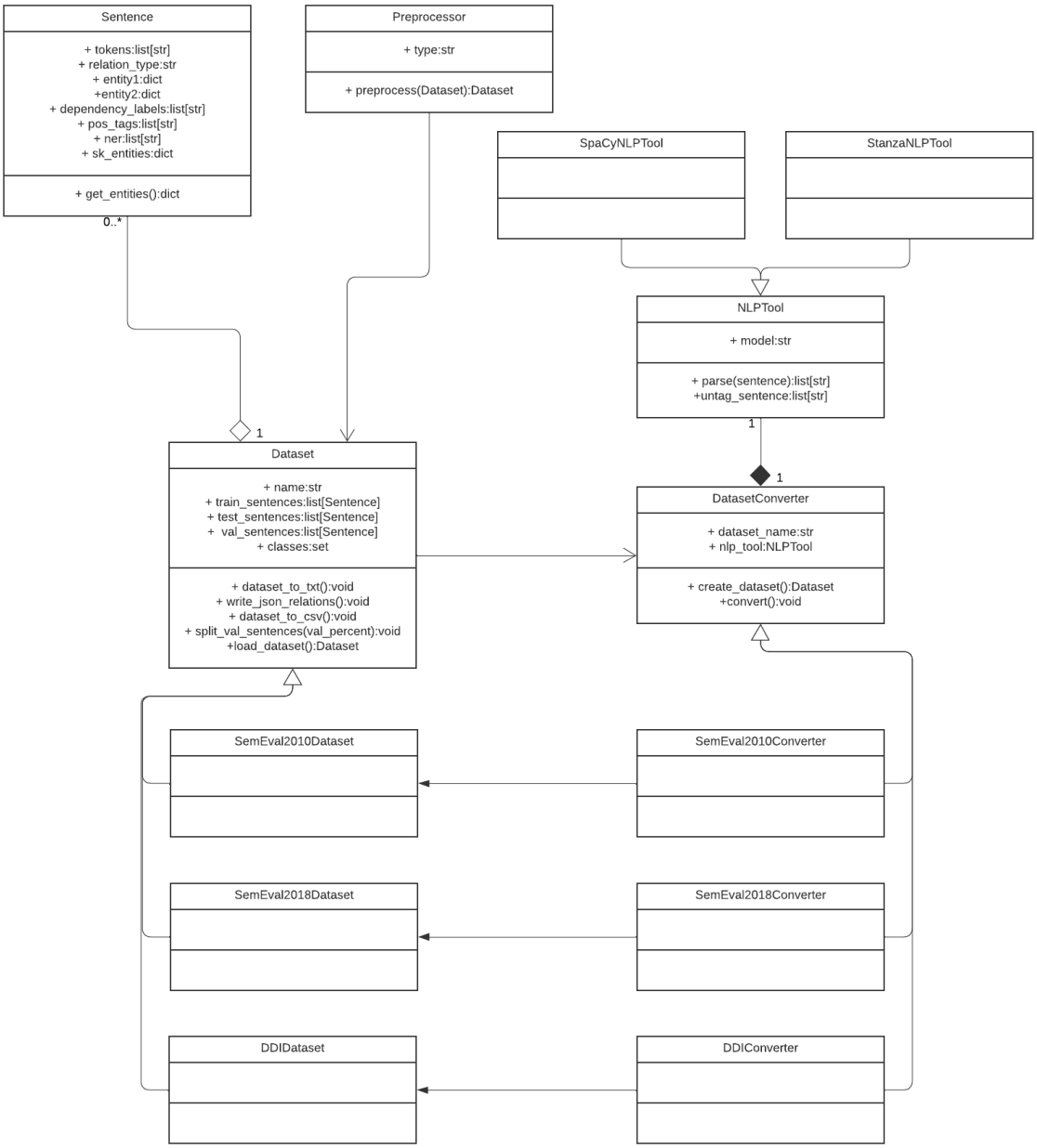

Figura 17 – Diagrama de classes do módulo PLN no *DeepREF*

ambas mudam as palavras da entidade com uma palavra mais genérica como *"ENTITY"* (para o *SemEval 2010* e *SemEval 2018*); e *"DRUG"* (para o *DDI 2013*). Ofuscamento de dígitos também muda o *token* numérico para uma palavra mais geral - "DIGIT". Além dos tipos de pré-processamento citados acima, existe a remoção de textos entre colchetes e parênteses da sentença.

Houve a criação de classes separadas para cada tipo de pré-processamento. Criou-se uma classe chamada *Preprocessor* que é a classe pai de todos os tipos de pré-processamentos,

que herdam os atributos e métodos da classe pai.

A figura 18 mostra o diagrama de classes do módulo PLN e de processamento de texto remodeladas a partir do *REflex*. Apresenta a classe pai *Preprocessor* que possui classes filhas para cada tipo de pré-processamento (*BracketsPreprocessor*, *EntityBlindingPreprocessor*, *DigitBlindingPreprocessor*, *PunctuationPreprocessor* e *StopWordsPreprocessor*). Existe um tipo de padrão de projeto no diagrama de classes de processamento de texto chamado de *template method* que é um padrão comportamental pois apresenta uma classe pai que implementa métodos que são comuns a todas as classes filhas, mas existe um método abstrato que é sobrescrito pelas classes filhas para serem implementados de acordo com a tarefa específica de cada um, que no caso são os diferentes tipos de pré-processamento disponíveis no *DeepREF*.

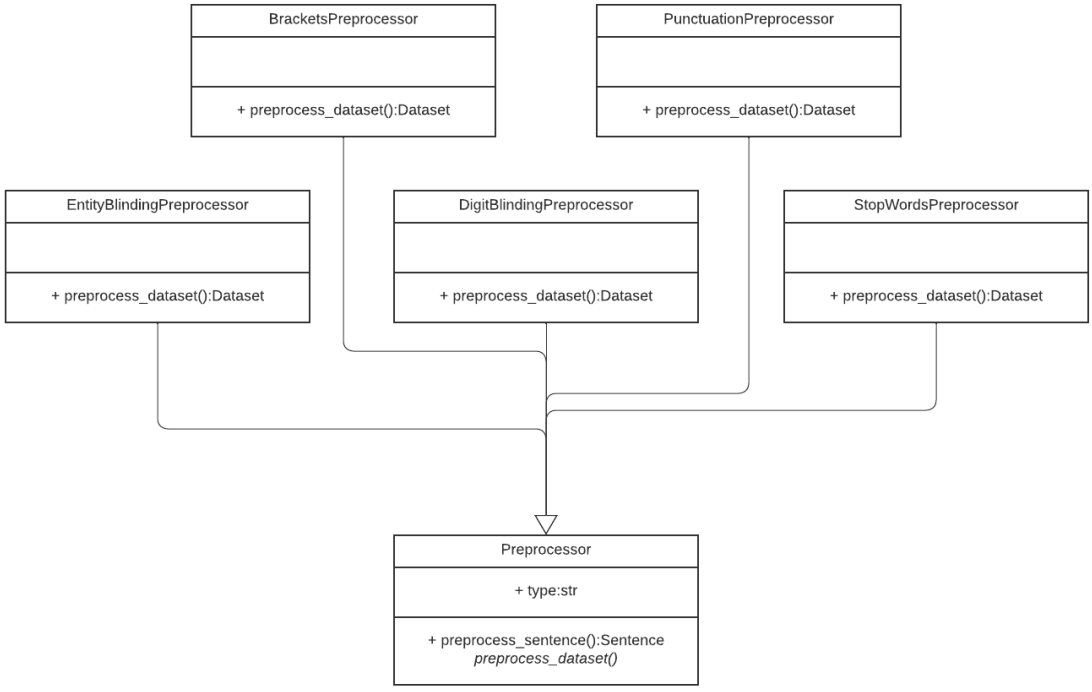

Figura 18 – Diagrama de classes do módulo de pré-processamento de texto no *DeepREF*

4.4 Módulo de encoding de sentença e token - Embeddings

Depois do passo de pré-processamento de texto, existe um módulo que pode realizar ambos *encoding* para sentença e *token* para gerar *embeddings* de texto.

Além do *embedding* BERT para sentença/*tokens*, *DeepREF* pode também produzir três outros tipos de *embeddings*: um *embedding* semântico, *embedding* de *part-of-speech (POS) tagging*, e *embedding* de rótulos de dependência (deps). Em particular, para o *embedding* semântico, os hiperônimos dos *tokens* que formam as entidades foram recuperados usando o *WordNet* (FELLBAUM, 1998) do NLTK (LOPER; BIRD, 2002).

***Embedding* de POS tags** é feito pelo extração dos *POS tags* de uma sentença, e gerando um embedding com a sequência de *POS tags*.

***Embeddings* de deps** é feito extraindo o grafo com a análise de dependência. Figura 19 mostra um exemplo de um grafo com análise de dependência. Assim, é possível obter todas as dependências em uma sentença e produzir um *embedding* com eles. Foi gerado os *embeddings* no framework de cada tipo acima utilizando-se a função *Embedding*[5] do *Pytorch*.

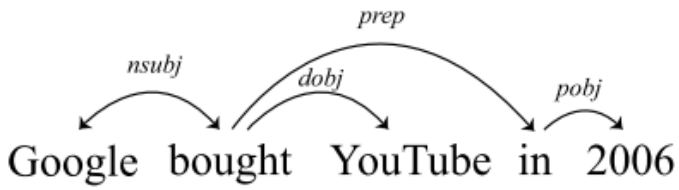

Figura 19 – Um exemplo de grafo de dependência (NASTASE et al., 2013).

Cada sentença no dataset tem uma sequência de *tokens*, i.e., $s = [w_1, ..., w_n]$, onde n é o tamanho da sentença. Cada *token* possui uma representação vetorial densa, i.e., $v = [v_1, ..., v_n] \in \mathcal{R}^{d_w \times n}$, onde d_w é a dimensão do *word embedding*.

Além do *word embedding*, foram usados *embeddings* de posição, semântica (*SK*), *POS tags* e *deps*. Para obter o *word embedding* das entidades, acessa-se as camadas finais escondidas correspondendo aos *tokens* das palavras em cada menção da entidade, o qual produz dois vetores v_{eh} e v_{et} que correspondem aos *word embeddings* das entidades *head e tail*, respectivamente. Os *embeddings* de posição, *SK*, *POS tags* and *deps* são, na verdade, as posições, *SK*, *POS tags* e *deps* das entidades *head e tail*, i.e., $\mathbf{r}^e_{pos}$, $\mathbf{r}^e_{sk}$, $\mathbf{r}^e_{pt}$, $\mathbf{r}^e_{deps}$, respectivamente, onde *pos*, *sk*, *pt*, *deps* denota posição, *SK*, *POS tags* and *deps* embeddings, respectivamente. O expoente e corresponde à entidade independentemente de ser entidades *head ou tail*.

[5] <https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html>

O *embedding* de posição é obtido extraindo o índice da primeira e segunda entidade na sentença. O *embedding SK* é obtido extraindo os hiperônimos (até o segundo nível na hierarquia do *WordNet*) relacionados à entidade. Por exemplo, a entidade "empresa" tem a palavra "instituição" como o mais próximo hiperônimo, que tem "organismo" como seu mais próximo hiperônimo. Então, "instituição" e "organismo" são os hiperônimos pai e avô, respectivamente, da entidade "empresa".

Os *embeddings de POS tags e deps* são obtidos extraindo os *POS tags* e os grafos de dependência de toda a sentença, usando as ferramentas PLN (*SpaCy* ou *Stanza*).

Considerou-se apenas os *embeddings de POS tags e deps* para a primeira e segunda entidade nesta etapa. Os *embeddings* são formalmente descritos nas seguintes equações para o modelo proposto no *DeepREF*:

$$s = [w_1, ..., w_{eh}, ..., w_{et}, ..., w_n, w^{'}_{skh}, w^{''}_{skh}, w^{'}_{skt}, w^{''}_{skt}] \tag{4.1}$$

$$v = [v_1, ..., v_{eh}, ..., v_{et}, ..., v_n, v^{'}_{skh}, v^{''}_{skh}, v^{'}_{skt}, v^{''}_{skt}] \tag{4.2}$$

$$\mathbf{r}^{eh} = [v_{eh}; v_{skh}; r^{eh}_{pos}; r^{eh}_{pt}; r^{eh}_{deps}] \tag{4.3}$$

$$\mathbf{r}^{et} = [v_{et}; v_{skt}; r^{et}_{pos}; r^{et}_{pt}; r^{et}_{deps}] \tag{4.4}$$

$$r^{e}_{embed} = \mathbf{W} \cdot \mathbf{X} + \mathbf{b} \tag{4.5}$$

$$\mathbf{r}^{e} = [\mathbf{r}^{eh}; \mathbf{r}^{et}] \tag{4.6}$$

onde w_{eh} e v_{eh} são a palavra da entidade-cabeça da sentença e o vetor correspondente, respectivamente; $w^{'}_{skh}$ e $v^{'}_{skh}$ são a palavra do hiperônimo pai da entidade-cabeça e seu vetor correspondente, respectivamente; e $w^{''}_{skh}$ e $v^{''}_{skh}$ são o hiperônimo avô da entidade-cabeça e seu vetor correspondente, respectivamente. A mesma regra vale para as palavras das

entidades-cauda ($w^{'}_{skt}$, $v^{'}_{skt}$, $w^{''}_{skt}$ e $v^{''}_{skt}$). O $r^{e}_{embed} \in \mathcal{R}^{d_e}$, d_e é a dimensão de r^{e}_{embed} e é igual a 5. O *embed* subscrito corresponde ao tipo do *embedding* e deve ser substituído pelos *embeddings* de posição, *POS tags* ou *deps*. $v_i \in \mathcal{R}^{d_w}$ and d_w é igual a 768. Então, $d_{r^{eh}} = d_{r^{et}} = 2 \times d_w + 3 \times d_e$ and $d_{r^e} = 2 \times d_{r^{eh}} = 2 \times d_{r^{et}}$, onde $d_{r^{eh}}$ é a dimensão da primeira entidade e $d_{r^{et}}$ é a dimensão da segunda entidade. As equações 4.3, 4.4 e 4.6 são concatenações dos *embeddings*.

Essas equações fazem parte de uma proposta de uma abordagem *entity-aware word embedding* enriquecida com *features* semânticas da primeira e segunda entidade, chamado E-BEM, que significa *enhanced BERT Entity Mention*, uma versão melhorada da versão do *BERT-EM* (SOARES et al., 2019). E-BEM é simplesmente a concatenação do *BERT-EM encoding* com o *SK embedding*. O E-BEM foi desenvolvido como um *baseline* para outros modelos que podem ser desenvolvidos no *DeepREF*.

A figura 20 mostra a simples arquitetura *Deep Learning* do *E-BEM* escolhida para os experimentos e utilizada como comparação em relação aos outros modelos das competições dos diferentes datasets e também dos modelos reproduzidos pelos *frameworks*.

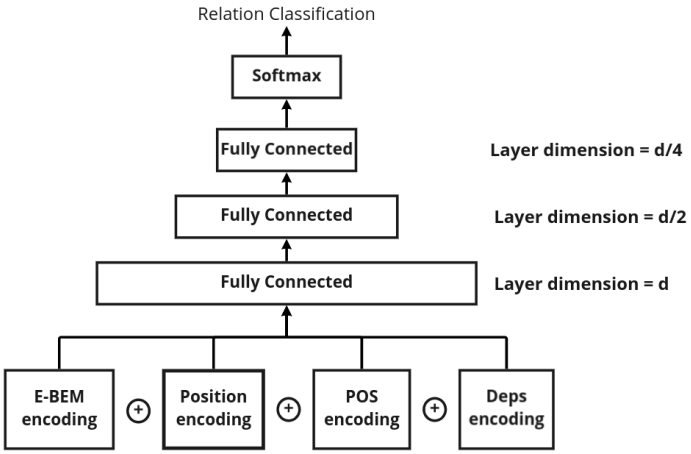

Figura 20 – Arquitetura do E-BEM usada nos experimentos.

4.5 Módulo Deep Learning - Otimização de hiperparâmetros

Depois do passo da encodificação de sentenças e *tokens*, o módulo de *Deep Learning* realiza o processo de aprendizagem usando redes neurais de aprendizagem profunda disponíveis no *DeepREF*, incluindo CNN, PCNN, CR-CNN, GRU/BiGRU, e

LSTM/BiLSTM. O processo de aprendizagem é realizado de acordo com a otimização de hiperparâmetros usando o *framework Optuna* (AKIBA et al., 2019). Assim, *DeepREF* tem algumas vantagens comparadas com outros *frameworks* que não empregam otimização de hiperparâmetros.

O *DeepREF* usa, por padrão no *Optuna*, algoritmos de otimização de hiperparâmetros estados-da-arte como *Tree Parzen Estimators* e *HyperBand*, que tem alcançado bons desempenhos ao encontrar os melhores hiperparâmetros (YU; ZHU, 2020).

Esses hiperparâmetros incluem o tamanho do *batch*, a taxa de aprendizado, o decaimento dos pesos[6], tamanho máxima da sentença e número máximo de épocas de treinamento.

Outros hiperparâmetros foram usados dos trabalhos de (ZENG et al., 2014; ZENG et al., 2015; SOARES et al., 2019) e são os mesmos hiperparâmetros utilizados por padrão no *OpenNRE*.

O código-fonte 4.1 mostra um exemplo necessário para executar uma sessão de treinamento no *DeepREF*. Note que é necessário escrever algumas linhas de código para treinar diferentes modelos considerando variáveis de experimentação distintas (*datasets*, tipos de pré-processamento, e tipos de *embedding*). Ao contrário do OpenNRE, a configuração dos hiperparâmetros do *DeepREF* é feita fora do código em um arquivo *json*, como mostrado no código fonte 4.2.

```
1 import json
2 from deepref import config
3 from deepref.framework.train import Training
4
5 with open(config.HPARAMS_FILE_PATH.format('semeval2010'), '
    r') as f:
6     hparams = json.load(f)
7
8 train = Training('semeval2010', 'micro_f1', hparams)
9 train.train()
```

Código-fonte 4.1 – Exemplo de código para treinamento do SemEval 2010 no DeepREF.

[6] Este hiperparâmetro é apenas otimizado para CNNs e RNNs no *DeepREF*, não para o BERT. O valor do decaimento de pesos é o padrão do OpenNRE para o BERT.

```
{
    "model": "ebem",
    "pretrain": "bert-base-uncased",
    "preprocessing": [],
    "batch_size": 16,
    "lr": 2e-05,
    "max_length": 128,
    "max_epoch": 3,
    "position_embed": 0,
    "pos_tags_embed": 0,
    "deps_embed": 0
}
```

Código-fonte 4.2 – Exemplo de configuração num arquivo *json* para treinamento de datasets no DeepREF.

Estes são os seguintes parâmetros usados pelo *DeepREF*: o modelo de rede neural, os *embeddings*, tipos de pré-processamentos e hiperparâmetros usados para treinamento. Diferente de (NASCIMENTO et al., 2022), o framework *DeepREF* foi simplificado para apenas treinar com os dados acima citados, ao invés de utilizar informações como a ferramenta de *tokenização* e análises sintáticas, tais como *SpaCy e Stanza*, que são escolhidas no momento do pré-processamento do *dataset*, no módulo processamento do presente *framework*.

O número "0" nos *position_ embed*, *pos_ tags_ embed* e *deps_ embed* significa que eles não serão considerados no treinamento do modelo E-BEM, que está indicado no *model* do arquivo *json*. Se o número for "1", significa que o *embedding* será considerado no treinamento do modelo. A configuração *preprocessing* possui como valor uma lista onde serão adicionados os tipos de pré-processamentos no dataset que se quer treinar. Ou seja, inserindo-se as siglas referentes aos tipos de pré-processamentos, está indicando que esses pré-processamentos serão realizados no dataset e treinados para o modelo. O restante dos números são os valores de cada hiperparâmetro. Note-se o pequeno número de linhas necessário, a inteligibilidade do código e a facilidade de se alterar a configuração de um

treinamento no *DeepREF* no código-fonte 4.1.

A tabela 9 mostra a distribuição dos hiperparâmetros utilizados na busca dos melhores hiperparâmetros no framework *Optuna*, além de mostrar os valores padrão utilizados no *DeepREF*. Esses valores padrão foram baseados no *OpenNRE*. O tamanho do *batch* é diferente do valor padrão do *OpenNRE* por conta da limitação do uso de memória RAM disponível na GPU utilizada para os experimentos.

Hiperparâmetros	**Valores padrão**	**Distribuição**
número de épocas	3	2-8
tamanho do batch	16	2-16
taxa de aprendizagem	2e-5	{1e-6, 0.1}
tamanho da sentença	128	16-256

Tabela 9 – Distribuições de hiperparâmetros para otimização de hiperparâmetros no último experimento. A distribuição escrita com {} significa que a distribuição é contínua. A distribuição que tem um "-" significa que é uma distribuição discreta entre um valor a outro e inclusivo.

4.6 Estudo de caso explicativo

Apresenta-se aqui um estudo de caso explicativo sobre a adição de um novo dataset, um novo pré-processamento e um novo modelo para mostrar a facilidade de utilização do *framework DeepREF*. O estudo de caso será um exemplo de acrescentar um dataset na língua portuguesa utilizando o *E-BEM* juntamente com um BERT da língua portuguesa e com o pré-processamento *Shortest Dependency-Path (SDP)*.

4.6.1 Criação de Novo Dataset

Para acrescentar o novo dataset em língua portuguesa, primeiramente é preciso adicionar o nome do dataset na variável *DATASETS* no arquivo de configurações do *framework* chamado de *config.py*.

Depois, é necessário adicionar um arquivo no formato seguinte: *nome-do-dataset_ converter.py*. Deve-se criar uma classe própria de conversor para esse dataset

herdando a classe *DatasetConverter* e sobreescrevendo apenas os métodos *get_sentences* e *get_entity_dict*. O método *get_sentences* é um gerador[7] em *Python* que retorna as sentenças, com suas respectivas *tags* que indicam textualmente na sentença quais são as entidades da relação binária, e a respectiva anotação da relação entre as entidades. O método *get_entity_dict* também é um gerador em *Python* que retorna um dicionário com as informações das entidades, tais como as posições de cada entidade e o texto que se refere à palavra correspondente às entidades. Pode-se usar mais métodos dependendo da complexidade do dataset e seu formato.

Após criar um novo arquivo "conversor" para a criação do dataset, ao executá-lo, é preciso colocar no comando o tipo de ferramenta PLN (*Spacy* ou *Stanza*) e o tipo de modelo que será utilizado para realizar a tokenização, o *parsing* de dependência e *POS tags*. Se for utilizada a ferramenta *Spacy*, é necessário baixar um modelo em português[8] para realizar as tarefas de PLN para o dataset de língua portuguesa. Da mesma forma utilizando a ferramenta *Stanza*, é preciso baixar o modelo de português para essa ferramenta[9]. Após baixar o modelo para uma das ferramentas, deve-se executar o comando em *Python* do "conversor" acrescentando-se as informações da ferramenta PLN a ser utilizada e o nome do modelo em português baixado.

Antes de executar o comando do "conversor", é necessário criar um diretório com o mesmo nome do dataset escolhido para a variável DATASETS no arquivo de configurações *config.py*. Pode-se utilizar o código *bash* como base para baixar o dataset no seu formato original e executar os comandos de criação de diretórios e conversão dos datasets para o formato padrão do *framework*. Os códigos em *bash* estão no diretório *benchmark* no formato *download_<nome-do-dataset>.sh*. Após executar o comando, haverá a conversão do formato original do dataset para o formato padrão do *framework* e o dataset estará disponível e pronto para ser treinado.

---

[7] Falando de forma simples, um gerador em Python é uma função que retorna um objeto iterador sobre o qual pode-se iterar um valor por vez.

[8] Para baixar um modelo em português é preciso seguir instruções no site do Spacy <https://spacy.io/models/pt>.

[9] Os modelos disponíveis no Stanza se encontram no site <https://stanfordnlp.github.io/stanza/available_models.html>.

4.6.2 Criação de Novo Pré-Processamento

Ao criar um novo tipo de pré-processamento de dataset, é preciso inserir uma nova sigla do pré-processamento que será utilizada para o treinamento no arquivo *config.py*. A sigla utilizada será *"sdp"*. É preciso também criar um novo arquivo *Python* no formato *<nome-do-pre-processamento>_preprocessor.py* no diretório *deepref/dataset/preprocessors/* do *framework*. Este novo arquivo possuirá uma classe que que herdará da classe pai *Preprocessor* presente no mesmo diretório.

O único método que a classe filha sobreescreverá da classe pai é o método *preprocess_dataset* que pega todas as sentenças de treino, teste e validação e pré-processa todas elas de acordo com o tipo de pré-processamento escolhido. No caso do exemplo do caso de uso, será utilizado o pré-processamento do *SDP* em cada sentença. Cria-se outro método para esse tipo de pré-processamento que pega uma sentença e retorna o *SDP* da mesma. Existe uma biblioteca chamada *Networkx*[10] que auxilia no tratamento de informações complexas transfomando-as em grafos, por exemplo. Pode ser utilizada em conjunto com as ferramentas *Spacy* ou *Stanza*.

Existe um método chamado *process_sentence* na classe pai *Preprocessor* que atualiza os índices das entidades de cada sentença para que não se perca a informação da posição da sentença. Essa informação é essencial para o treinamento dos modelos. Caso algum pré-processamento ocasionar uma diminuição ou aumento na quantidade de *tokens* da sentença, será necessário utilizar esse método para atualização das posições das entidades na sentença.

4.6.3 Treinamento com BERT pré-treinado para língua português-brasileira

Essa etapa é a do treinamento do *E-BEM* utilizando-se o BERT de língua portuguesa, especificamente do português brasileiro. Existe um BERT pré-treinado para a língua português-brasileira chamado de *BERTimbau* (SOUZA et al., 2020). Assim como para o BERT da língua inglesa, o *BERTimbau* alcança resultados estados-da-arte em várias tarefas PLN (SOUZA et al., 2020) para o português brasileiro. O *BERTimbau* é livre e disponível para uso público a partir do *Hugging Faces*[11]. No site do *Hugging Faces*

[10] <https://networkx.org/documentation/stable/index.html>
[11] <https://huggingface.co/neuralmind/bert-base-portuguese-cased>.

explica-se como se utiliza o *BERTimbau*.

É necessário criar um arquivo no formato *hyperparams_ <nome-do-dataset>.json* no diretório *hyperparameters* com as configurações dos hiperparâmetros e outros dados como o tipo de pré-processamento realizado no dataset para treinamento, quais tipos de *embeddings* serão utilizados em conjunto com os dados de treinamento (*embeddings* de posição, *POS tags* e dependência), qual o modelo e os pesos pré-treinados dos *word embeddings*. O código-fonte 4.3 mostra um exemplo de configuração para treinamento de classificação de relações utilizando-se dataset de língua portuguesa, com o modelo *E-BEM* juntamente com o *BERTimbau* e pré-processamento *SDP*.

```
{
    "model": "ebem",
    "pretrain": "bert-base-portuguese-cased",
    "preprocessing": ['sdp'],
    "batch_size": 16,
    "lr": 2e-05,
    "max_length": 128,
    "max_epoch": 3,
    "position_embed": 0,
    "pos_tags_embed": 0,
    "deps_embed": 0
}
```

Código-fonte 4.3 – Exemplo de configuração num arquivo *json* para treinamento de um dataset em língua português-brasileiro no DeepREF.

5 Avaliação Experimental

Em primeiro lugar, escolheu-se o modelo que possui melhor resultado no *framework*, como está no artigo do OpenNRE (HAN et al., 2019) e o modelo escolhido foi o *BERT-EM* que, adicionado com o *embedding sk*, tornou-se o *E-BEM*. Depois, foram realizados vários experimentos com todas as combinações de pré-processamento e *embeddings* para cada dataset. O melhor resultado foi escolhido e foram realizados vários experimentos de otimização utilizando-se o *framework Optuna* para encontrar os melhores valores de hiperparâmetros para cada *dataset*. Os experimentos foram executados utilizando-se a GPU NVIDIA GeForce GTX 1660 Ti Mobile com 6GB de capacidade de memória RAM.

5.1 Configurações dos melhores modelos E-BEM

Os experimentos utilizando todas as combinações de tipos de pré-processamento e *embeddings* foram realizados utilizando-se os valores padrão da tabela 9. Depois desses experimentos foi possível verificar qual combinação resulta no melhor desempenho. As melhores combinações de tipos de pré-processamento e *embeddings* pode ser visto na tabela 10.

Dataset	**Tipo de pré-processamento**	**Embeddings**
SemEval 2010	b	-
SemEval 2018 ST1	b+d+p	deps
SemEval 2018 ST2	d+sw	pos+deps
DDI	eb	pos+deps

Tabela 10 – A melhor combinação de tipo de pré-processamento e embedding para cada *dataset* avaliado. d = ofuscamento de dígitos; b = remoção de colchetes e parênteses; p = remoção de pontuação; sw = remoção de *stopwords*; eb = ofuscamento de entidades; pos = embedding de POS tags; deps = embedding de dependência.

Importante salientar que no *framework DeepREF*, assim como no *OpenNRE*, a inicialização dos pesos do modelo é sempre a mesma para que não haja discrepância nos resultados devido à aleatoriedade da inicialização dos pesos dos modelos de DL. Dessa

forma, é possível comparar os diferentes modelos e arquiteturas com mesma inicialização dos pesos.

Diante do resultado da tabela 10, pode-se ver que a metade dos *datasets* avaliados produzem melhores resultados com o pré-processamento de remoção de texto entre colchetes ou parênteses e outra metade se beneficia do ofuscamento de dígitos. Isso se deve ao fato de que tais pré-processamentos removem informações com "ruído" das sentenças.

5.2 Estudos de ablação

O *SemEval 2010 Task-8* é o dataset mais simples em termos de tamanho de sentença, como se pode ver no gráfico da figura 21, e utilização de palavras mais genéricas e comuns, dentre os datasets avaliados. Esse é o provável motivo de que os pré-processamentos executados sobre esse dataset não façam muito efeito, pois a ideia do pré-processamento é simplificar a sentença eliminando os "ruídos" que possam dificultar a classificação correta das relações. Outras figuras de gráficos de quantidades de sentença por tamanho para os outros datasets se encontram no Apêndice A.

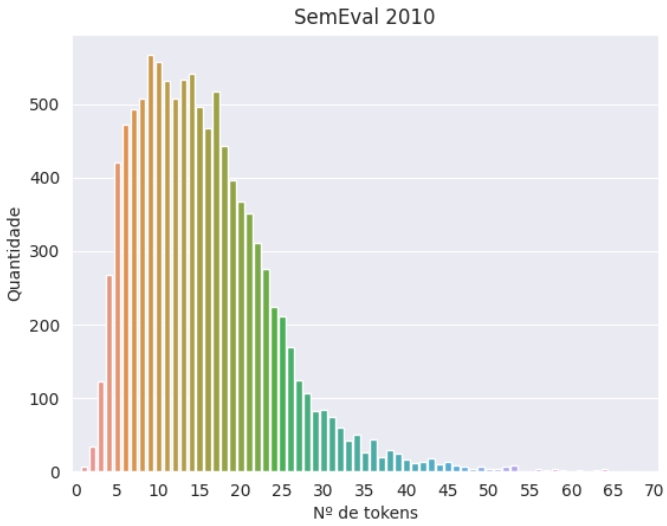

Figura 21 – Estatística da quantidade de sentenças por tamanho para o dataset SemEval 2010.

Para o *SemEval 2010*, obteve-se uma melhoria pequena de aproximadamente 0,21 pontos (0,23%) em relação ao resultado do dataset sem nenhum tipo de pré-processamento na métrica Micro-F1. Porém, para o Macro-F1, o dataset *SemEval 2010* sem

pré-processsamento apresenta melhor resultado do que com pré-processamento de ofuscamento de dígitos com aproximadamente a mesma porcentagem da melhoria na métrica Micro-F1. A tabela 11 mostra os 5 melhores resultados para o dataset *SemEval 2010* utilizando o E-BEM.

Features	**Pré-processamento**	**Micro-F1**	**Macro-F1**
E-BEM	b	**88,64**	81,21
E-BEM	-	88,42	**81,40**
E-BEM+p+deps	b	88,39	80,93
E-BEM+pos+deps	b	88,36	81,08
E-BEM+p+pos	d	88,33	80,72

Tabela 11 – Os 5 melhores resultados para o dataset SemEval 2010 sem otimização.

O *SemEval 2010* apresenta melhores resultados sem *embeddings* no E-BEM, porém, já possuindo apenas o *embedding* semântico (*sk*) contido no E-BEM. Isso evidencia que esse *embedding* já é suficiente para melhorar o desempenho do modelo para o *SemEval 2010* pois acrescenta informações para as entidades de uma sentença. Será explorado, como trabalho futuro, o motivo dos *embeddings de POS tags e deps* não trazerem melhores resultados para o SemEval 2010. Uma hipótese é de que melhorando a arquitetura do modelo (adicionando uma GNN, por exemplo) estruturaria melhor a adição de informação de *POS tags e deps* e, dessa forma, melhoraria os resultados utilizando-se esses *embeddings*.

O dataset *SemEval 2018* ST1 apresenta melhores resultados com 3 tipos de pré-processamentos (remoção de palavras entre colchetes ou parênteses, ofuscamento de dígitos e remoção de pontuação). Neste dataset existem muitas palavras entre parênteses e a remoção de palavras entre parênteses ajuda o modelo a não focar em palavras que não são úteis para a classificação da relação de entidades. As pontuações também acrescentam "ruídos" nas sentenças e dificulta a classificação de relações. Um tipo de pré-processamento que é comum para as subtarefas ST1 e ST2 do *SemEval 2018* é o ofuscamento de dígitos. Isso acontece pois há muitos números nestes datasets que não são relevantes para a tarefa de classificação de relações e apenas acrescenta "ruído" ao modelo. A tabela 12 mostra os melhores resultados para o dataset *SemEval 2018 ST1*.

Além de se beneficiar do pré-processamento de ofuscamento de dígitos, o *SemEval*

Features	**Pré-processamento**	**Micro-F1**	**Macro-F1**
E-BEM+deps	b+d+p	**86,48**	**87,09**
E-BEM+p+deps	b+d	85,92	85,82
E-BEM+pos+deps	p	84,23	84,91
E-BEM+p+deps	d	84,79	84,23
E-BEM+pos	b	84,51	84,22

Tabela 12 – Os 5 melhores resultados para o dataset SemEval 2018 ST1 sem otimização.

2018 Task-7 ST2 teve melhores resultados com o tipo de pré-processamento de remoção de *stop words* pois apresenta muitas sentenças com grande tamanho e esse tipo de pré-processamento remove as palavras *stop words*, que são as palavras mais comuns em qualquer linguagem (artigos, preposições, pronomes e conjunção) e não adicionam muita informação ao texto. Apenas esse dataset apresenta os melhores resultados com o tipo de pré-processamento de remoção de *stop-words*. Isso se deve provavelmente porque alguns *stop words* são importantes para identificação da relação em outros datasets como, por exemplo, *because*, que é um *stop word* que pode indicar relações de causa, e *about*, que é um *stop word* que pode indicar relações sobre tópicos. A tabela 14 mostra os melhores resultados para o dataset *SemEval 2018 ST2*.

O dataset DDI Extraction 2013 se beneficia mais do pré-processamento de ofuscamento de entidades porque esse tipo de pré-processamento troca as entidades numa sentença para uma palavra mais geral como o *"DRUG"*. O *DDI Extraction 2013* apresenta em suas sentenças nomes de medicamentos que não estão presentes na linguagem comum do Inglês e, por isso, o ofuscamento de entidades apresenta melhores desempenhos para este dataset pois os nomes dos medicamentos adicionam "ruído" ao modelo e não são importantes na classificação. Ou seja, mudando o nome do medicamento para *"DRUG"* faz com que o modelo foque nas relações entre as entidades e não nos nomes dos medicamentos. O ofuscamento de entidades nos outros datasets não funcionou bem por conta da importância do conhecimento das entidades para a classificação de relação na sentença. O DDI 2013 obteve os melhores resultados utilizando o pré-processamento de ofuscamento de entidades como se pode ver na tabela 13.

Todos os outros datasets, exceto o *SemEval 2010*, obtiveram os melhores resultados

Features	**Pré-processamento**	**Micro-F1**	**Macro-F1**
E-BEM+pos+deps	eb	**93,40**	**87,38**
E-BEM+pos+deps	eb+sw	93,40	87,38
E-BEM+pos+deps	d+eb	93,40	87,38
E-BEM+pos+deps	b+eb	93,40	87,38
E-BEM+pos+deps	p+eb	93,40	87,38

Tabela 13 – Os 5 melhores resultados para o dataset DDI 2013 sem otimização.

Features	**Pré-processamento**	**Micro-F1**	**Macro-F1**
E-BEM+pos+deps	d+sw	89,58	**90,63**
E-BEM	d+p	**89,86**	90,49
E-BEM+p+pos+deps	d	87,61	89,72
E-BEM+p+pos+deps	b+d+p	87,61	89,72
E-BEM+p+pos+deps	b+d+p+sw	87,61	89,72

Tabela 14 – Os 5 melhores resultados para o dataset SemEval 2018 ST2 sem otimização.

utilizando *embeddings* de *deps*. E a metade dos datasets, além de se beneficiar do *embedding de deps*, beneficia-se também do *embedding* de *POS tags*. Isso evidencia a importância desses dois *embeddings* no resultado de classificação de relações. Foram utilizadas as apenas as métricas de Micro-F1 e Macro-F1 pois são as métricas mais utilizadas por outros modelos estados-da-arte e servem, por isso, para fins de comparação com outros modelos.

Quando combinado com o pré-processamento e os *embeddings* corretos, o modelo consegue atingir os melhores resultados do que sem nenhum tipo de pré-processamento e *embedding*. Observa-se que os ganhos ao combinar os tipos de pré-processamento e *embeddings* corretos podem ser grandes como, por exemplo, o DDI 2013 que teve uma melhoria de 1,64% comparando o melhor modelo utilizando-se a melhor combinação de pré-processamento e *embedding* com o modelo sem nenhum tipo de pré-processamento e *embedding*; e uma melhoria de 3,17% comparando o melhor modelo da melhor combinação com o modelo com apenas o melhor pré-processamento. O *SemEval 2018 ST1 e ST2* também tiveram resultados significativos em relação ao modelo sem *embeddings* e pré-processamento. Interessante notar que alguns modelos tem resultados ruins

utilizando-se apenas o pré-processamento ou apenas os *embeddings*, mas a combinação desses dois parâmetros resulta numa significativa melhoria em todos os modelos, até mesmo para o dataset *SemEval 2010* pois o E-BEM já é um modelo embutido com o *embedding* semântico (*sk*). Além do mais, o E-BEM no *SemEval 2010* ganha nos resultados, mesmo sem pré-processamento, contra os resultados do BERT-EM do *OpenNRE* (HAN et al., 2019). Os estudos de ablação dos resultados das melhores combinações de cada dataset avaliado encontram-se na tabela 15. As métricas utilizadas foram Micro-F1 e Macro-F1 pois são as métricas utilizadas pelas competições dos datasets.

Dataset	**Features**	**Pré-processamento**	**Métrica**	**Resultado**	**Δ**
SemEval 2010	E-BEM	-	Micro-F1	88,42	-
	E-BEM	b	Micro-F1	**88,64**	0,24
SemEval 2018 ST1	E-BEM	-	Macro-F1	82,02	-
	E-BEM	b+d+p	Macro-F1	80,51	-1,51
	E-BEM+deps	-	Macro-F1	82,25	0,23
	E-BEM+deps	b+d+p	Macro-F1	**87,09**	5,07
SemEval 2018 ST2	E-BEM	-	Macro-F1	85,07	-
	E-BEM	d+sw	Macro-F1	77,84	-7,23
	E-BEM+pos+deps	-	Macro-F1	84,75	-0,32
	E-BEM+pos+deps	d+sw	Macro-F1	**90,63**	5,56
DDI	E-BEM	-	Micro-F1	91,89	-
	E-BEM	eb	Micro-F1	90,53	-1,36
	E-BEM+pos+deps	-	Micro-F1	93,06	1,17
	E-BEM+pos+deps	eb	Micro-F1	**93,40**	1,51

Tabela 15 – Estudos de ablação para os datasets SemEval 2010, SemEval 2018 e DDI 2013 para as melhores combinações de pré-processamento e *embeddings*. p = embeddings de posição; pos = embeddings de POS tags; deps = embeddings de dependência; d = ofuscamento de dígitos; b = remoção de palavras entre parênteses; sw = remoção de *stop words*; eb = ofuscamento de entidades; p = remoção de pontuações.

Observa-se nas tabelas dos 5 melhores resultados do DDI 2013 (Tabela 13) e SemEval

2018 ST2 (Tabela 14), que apresentam mesmos resultados para mesmas configurações de *embeddings* mas diferentes pré-processamentos.

Após encontrar a melhor combinação de tipo de pré-processamento e *embedding* que traz o melhor resultado das métricas de DL sobre cada *dataset* (Micro-F1 para os *datasets SemEval 2010 Task-8 e DDI Extraction 2013* e Macro-F1 para o *SemEval 2018 Task-7*), utilizou-se essa configuração para encontrar os hiperparâmetros que otimizam os resultados. A tabela 16 mostra as melhores combinações de hiperparâmetros resultante das 100 tentativas realizadas pelo framework de otimização de hiperparâmetros *Optuna*.

A tabela 17 resume os estudos de ablação em cada dataset para comparar a melhoria que cada *embedding* traz para o modelo. Os resultados mostrados foram para os datasets sem nenhum tipo de pré-processamento. Pode-se ver pela tabela que os *embeddings* não produzem bons resultados para os *datasets* sem algum tipo de pré-processamento. É necessário combinar algum tipo de pré-processamento no dataset para que haja uma melhoria nos resultados como se pode ver na tabela 15.

5.3 Comparação com estados-da-arte

As tabelas 18-21 mostram os resultados com e sem otimização de hiperparâmetros nos *datasets* SemEval 2010, SemEval 2018 ST1, SemEval 2018 ST2 e DDI Extraction 2013, respectivamente.

Uma olhada mais de perto nos resultados mostra que, para todos os *datasets*, *DeepREF* também superou os outros sistemas comparados ou tem resultados competitivos, exceto para o *dataset SemEval 2010*. Contudo, quando se foca apenas nos frameworks, *DeepREF* obteve os melhores modelos de aprendizado no *dataset SemEval 2010* (comparado com *OpenNRE* e *REflex*) e no *dataset DDI Extraction 2013* (comparado com o *REflex*). Em geral, a simples arquitetura do E-BEM mostra resultados encorajadores desde que este foi o melhor modelo nos *datasets SemEval 2018 Task-7 e DDI 2013*, mostrando que o processo de aprendizado com *embeddings*, pré-processamento e otimização de hiperparâmetros parece ser efetivo. Por outro lado, os melhores desempenhos de sistemas no *dataset SemEval 2010* foram implementados usando redes neurais profundas mais complexas tais como CNN, GNN, e RNN com mecanismo de atenção. Ao contrário, o E-BEM consiste em uma arquitetura MLP com 3 camadas escondidas.

Hiperparâmetro	**Melhor configuração encontrada**
SemEval 2010 Task-8	
número de épocas	3
tamanho do batch	16
tamanho da sentença	128
taxa de aprendizado	2,9267310071717223e-5
SemEval 2018 Task-7 ST1	
número de épocas	3
taamanho do batch	16
tamanho da sentença	128
taxa de aprendizado	9,443809143195011e-6
SemEval 2018 Task-7 ST2	
número de épocas	7
tamanho do batch	2
tamanho da sentença	164
taxa de aprendizado	1,3402454555251181e-5
DDI Extraction 2013	
número de épocas	6
tamanho do batch	2
tamanho da sentença	251
taxa de aprendizado	5,894304290161342e-6

Tabela 16 – Melhor configuração de hiperparâmetros depois da otimização de hiperparâmetros com 100 tentativas em cada *dataset*.

Como se pode ver também nos resultados das tabelas 18-21, a otimização de hiperparâmetros pode ser de grande valor para a melhoria dos resultados dos modelos. Alguns modelos não obtiveram uma melhoria expressiva utilizando a otimização de hiperparâmetros, mas o *SemEval 2018 ST2* conseguiu um ganho significativo de 1,7 pontos (aproximadamente 1,9%) de Macro-F1. Os outros resultados talvez não tenham sido tão expressivos pois o otimizador de hiperparâmetros não tenha convergido para um

Dataset	Features	Acurácia	Micro-F1	Macro-F1
SemEval 2010 Task-8	E-BEM	**84,32**	**88,42**	**81,40**
	E-BEM+p	84,25	88,40	81,21
	E-BEM+p+pos	84,14	87,77	79,69
	E-BEM+p+pos+deps	82,85	86,88	79,01
SemEval 2018 Task-7 ST1	E-BEM	**84,79**	**84,79**	**82,02**
	E-BEM+p	84,23	84,23	78,43
	E-BEM+p+pos	83,38	83,38	79,37
	E-BEM+p+pos+deps	80,28	80,28	77,90
SemEval 2018 Task-7 ST2	E-BEM	**91,55**	**91,55**	**85,03**
	E-BEM+p	87,89	87,89	84,92
	E-BEM+p+pos	87,04	87,04	81,11
	E-BEM+p+pos+deps	89,30	89,30	87,93
DDI Extraction 2013	E-BEM	88,87	91,89	84,15
	E-BEM+p	**90,81**	**92,96**	**87,04**
	E-BEM+p+pos	88,87	91,76	84,51
	E-BEM+p+pos+deps	89,99	92,17	85,59

Tabela 17 – Estudos de ablação para os datasets SemEval 2010, SemEval 2018 e DDI 2013 usando o E-BEM sem pré-processamento. p = embeddings de posição; pos = embeddings de POS tags; deps = embeddings de anotações de dependência.

valor significativo.

5.4 Otimização de hiperparâmetros

Pode-se observar, pelos gráficos das figuras 22 e 23, que removendo alguns parâmetros como o tamanho do *batch* e o número máximo de épocas — tornando-os fixos — ajuda numa convergência mais rápida, pois quanto mais parâmetros utilizados na otimização, maior a complexidade de tempo como descrito na Seção 2.6.

A maioria dos experimentos evidenciam que o hiperparâmetro da taxa de aprendizado influencia muito na melhoria dos resultados. Ao contrário dos hiperparâmetros

Modelo	**Micro-F1**
QA (COHEN et al., 2020)	**91,90**
RIFRE (ZHAO et al., 2021)	91,30
REDN (LI; TIAN, 2020)	91,00
Skeleton-Aware BERT (TAO et al., 2019)	90,36
BERT-EM+MTB (SOARES et al., 2019)	89,50
BERT-EM (SOARES et al., 2019)	89,20
E-BEM Otimizado (Nosso)	88,80
E-BEM (Nosso)	88,64
BERT-EM - OpenNRE (HAN et al., 2019)	88,30
BERT-tokens - REflex (CHAUHAN et al., 2019)	86,69

Tabela 18 – Resultados de *performance* em termos de Micro-F1 dos *modelos* no dataset SemEval 2010 Task-8.

Modelo	**Macro-F1**
E-BEM Otimizado (Nosso)	**87,30**
E-BEM (Nosso)	87,09
ETH-DS3Lab (ROTSZTEJN et al., 2018)	81,72
UWNLP (LUAN et al., 2018)	78,90
SIRIUS-LTG-UiO (NOORALAHZADEH et al., 2018)	76,70
DMIR (HETTINGER et al., 2018)	74,89
Talla (PRATAP et al., 2018)	74,20

Tabela 19 – Resultados de *performance* em termos de Macro-F1 dos modelos no *dataset* SemEval 2018 Task-7 ST1.

como o tamanho da sentença, o número de épocas e tamanho do *batch* que apresentaram valores pequenos de importância segundo os dados do gráfico da figura 22. O gráfico da figura 23 já mostra o inverso da figura 22 pois mostra que o hiperparâmetro da taxa de aprendizado como o menor valor. Isso é devido ao fato de que alguns valores de hiperparâmetro decair bastante o desempenho do modelo como o tamanho do *batch* e o tamanho da sentença, que são diretamente proporcionais à capacidade da memória

Modelo	**Macro-F1**
E-BEM Otimizado (Nosso)	**92,33**
E-BEM (Nosso)	90,63
ETH-DS3Lab (ROTSZTEJN et al., 2018)	90,40
Talla (PRATAP et al., 2018)	84,80
SIRIUS-LTG-UiO (NOORALAHZADEH et al., 2018)	83,20
MIT-MEDG (JIN et al., 2018)	80,60
GU IRLAB (MACAVANEY et al., 2018)	78,90

Tabela 20 – Resultados de *performance* em termos de Macro-F1 dos modelos no *dataset* SemEval 2018 Task-7 ST2.

Modelo	**Micro-F1**
E-BEM Otimizado (Nosso)	**93,99**
E-BEM (Nosso)	93,40
BERT-tokens - REflex (CHAUHAN et al., 2019)	91,31
DESC+MOL+SciBERT (ASADA et al., 2020)	84,08
SciFive-large (PHAN et al., 2021)	83,67
CharacterBERT (BOUKKOURI et al., 2020)	80,60
MOL+CNN (ASADA et al., 2018)	72,55

Tabela 21 – Resultados de *performance* em termos de Micro-F1 dos modelos no *dataset* DDI Extraction 2013.

da GPU, e, a partir de certo tamanho, podem ocasionar extrapolação da memória da GPU. Isso retorna o pior resultado para o modelo. Esses dois hiperparâmetros (tamanho da sentença e tamanho do *batch*) têm um caráter mais de auxiliar o desempenho do modelo do que melhorar os resultados. Dessa forma, parece que a taxa de aprendizado é o hiperparâmetro mais importante na melhoria dos resultados do modelo. A tabela 16 mostra que na combinação dos hiperparâmetros, os datasets *SemEval 2010* e *SemEval 2018 ST1* apresentaram os valores padrão para o tamanho da sentença, tamanho do *batch* e número de épocas corroborando, assim, que o hiperparâmetro mais importante é a taxa de aprendizado.

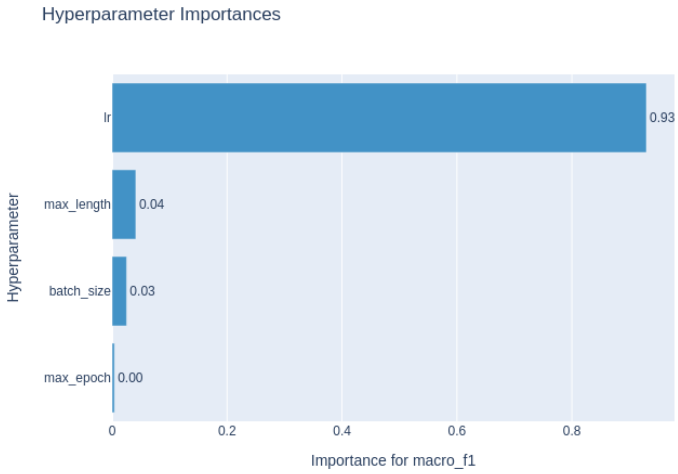

Figura 22 – Gráfico a importância de cada hiperparâmetro para o Macro-F1 durante otimização no SemEval 2018 ST1.

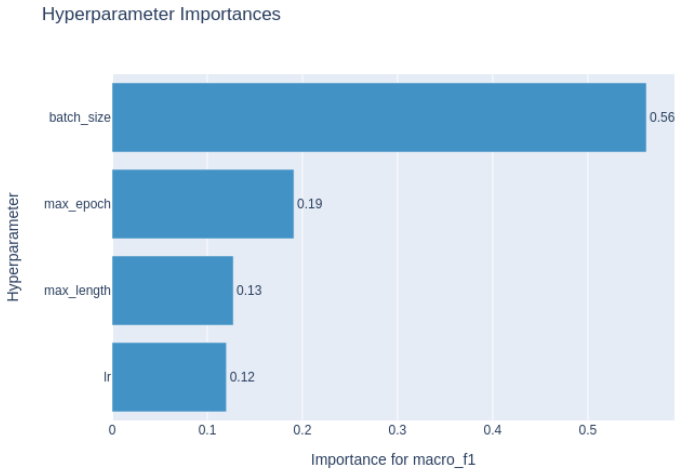

Figura 23 – Gráfico a importância de cada hiperparâmetro para o Macro-F1 durante otimização no SemEval 2018 ST2.

Os gráficos das figuras 24 e 25 mostram o intervalo de valores dentre os hiperparâmetros que obtiveram ganhos nos resultados. As linhas azuis mais fortes são os experimentos que tiveram melhores resultados. De modo geral, o que esses gráficos querem mostrar é que para o tamanho da sentença, os melhores valores estiveram entre 16 à 150; a maioria dos melhores valores dos modelos otimizados utilizaram a partir de 3 épocas; o melhor intervalo para a taxa de aprendizado foi entre 10^{-4} e 10^{-6}; e os melhores valores para o tamanho do *batch* variou bastante entre 2 e 16.

Esses valores são importantes para uma melhor escolha de intervalos num campo

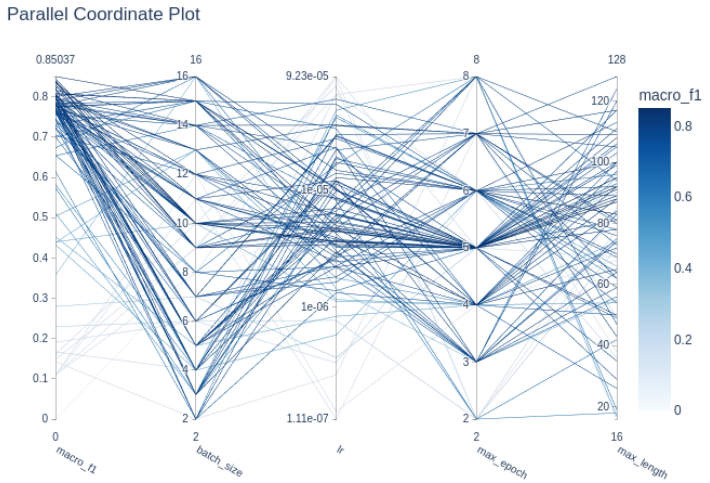

Figura 24 – Gráfico de coordenadas paralelas obtidos durante otimização de hiperparâmetros no SemEval 2018 ST1.

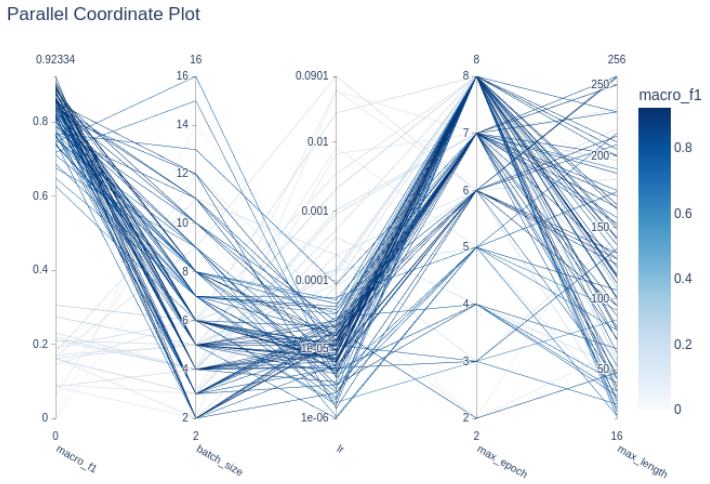

Figura 25 – Gráfico de coordenadas paralelas obtidos durante otimização de hiperparâmetros no SemEval 2018 ST2.

de busca menor, a fim de convergir para um resultado melhor mais rapidamente.

5.5 Matrizes de confusão

A matriz de confusão para o *SemEval2010* apresenta muita confusão com a classe *Other*. Isso se deve ao fato de que essa classe possui a maioria dos exemplos no dataset. Isto é, quase 20% dos exemplos de treino e de teste são da classe *Other* de acordo com a tabela 4. A matriz de confusão do *SemEval 2010* se encontra na tabela 26.

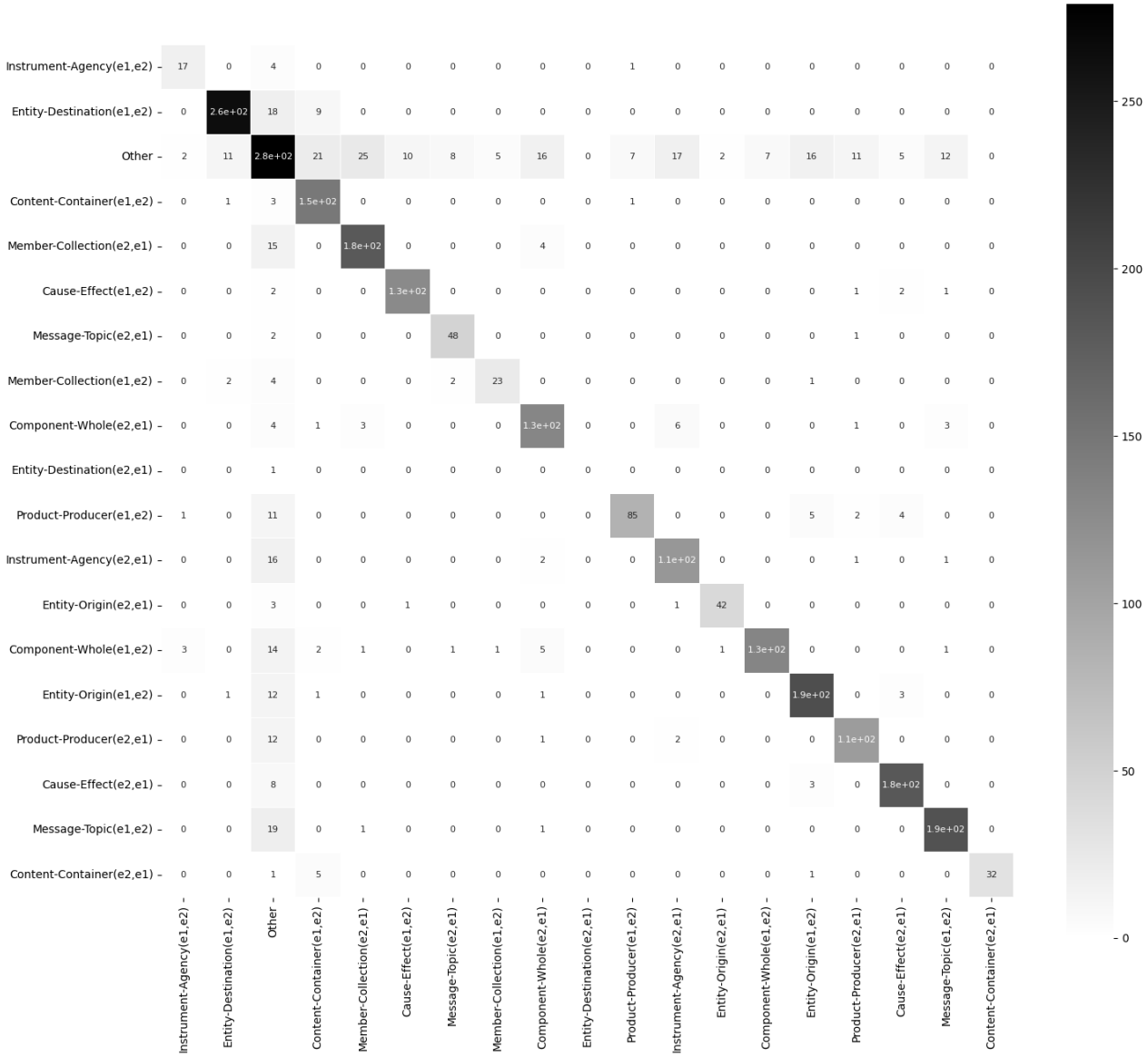

Figura 26 – Matriz de confusão para o melhor resultado do SemEval 2010.

As matrizes de confusão para o dataset *SemEval 2018* tanto ST1 e ST2 (tabelas 27 e 28, respectivamente) mostram que as classes em que os modelos mais confundem são as classes *part-whole* e *model-feature*. Essas classes confundem muito entre si e entre a classe *usage*. A confusão que o modelo faz com o *usage* se deve ao fato de haver mais exemplos de sentenças com entidades os quais apresentam relação de *usage* como descrito na tabela 5. Portanto, o modelo apresenta o viés de gerar como resultado a classe *usage* pois é a classe que mais possui exemplos no *SemEval 2018* tanto no ST1 quanto no ST2.

As classes *model-feature* e *part-whole* apresentam quantidades e porcentagens semelhantes no dataset e, também, palavras iguais mas com semânticas diferentes (como a preprosição em inglês *of*), por isso, que confundem-se entre si. Como exemplo, existe uma sentença em que entre a primeira e segunda entidade é da seguinte forma: "*corpus of spontaneous spoken dialogue*" em que *spontaneous spoken dialogue* é a primeira entidade e *corpus* é a segunda entidade. O tipo de relação entre eles é *part-whole*. Já outra sentença com "*informativeness of a word*" com *informativeness* como a primeira entidade e *word* como a segunda entidade e a relação entre elas é de *model-feature*. Ambas possuem mesma palavra com a mesma função na sentença que é de preposição. Por isso também que o *embedding* de *POS tags* não ajuda muito o modelo nesse caso, pois ambos possuem as mesmas informações e, portanto, mesmos *embeddings* de *POS tags* para a preprosição *of*.

A matriz de confusão do DDI 2013 (tabela 29) mostra que a classe *int*, que significa nenhuma reação medicamental, confunde bastante com a classe *effect*. Além da classe *int* ser a classe com menos exemplo no dataset DDI 2013, de acordo com a tabela 6, a classe *effect* é a classe com mais exemplos. Dessa forma, há um viés para a classe *effect* por conta da sua quantidade de exemplos.

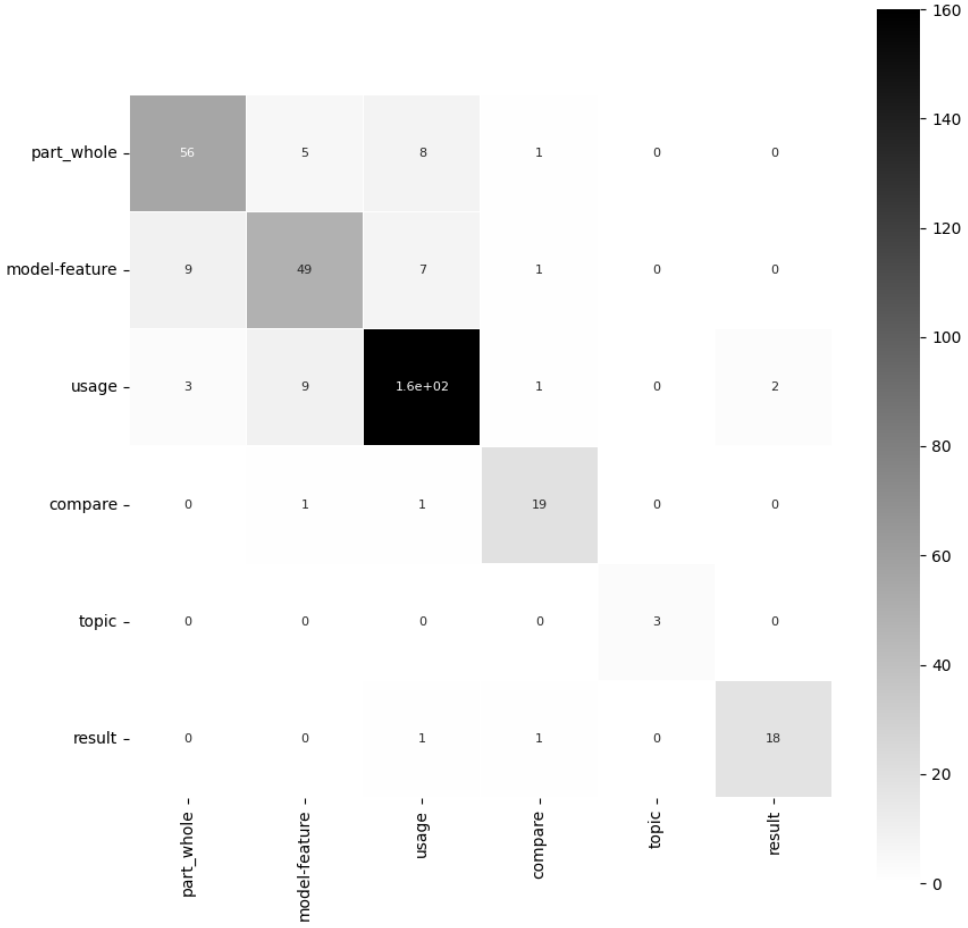

Figura 27 – Matriz de confusão para o melhor resultado do SemEval 2018 ST1.

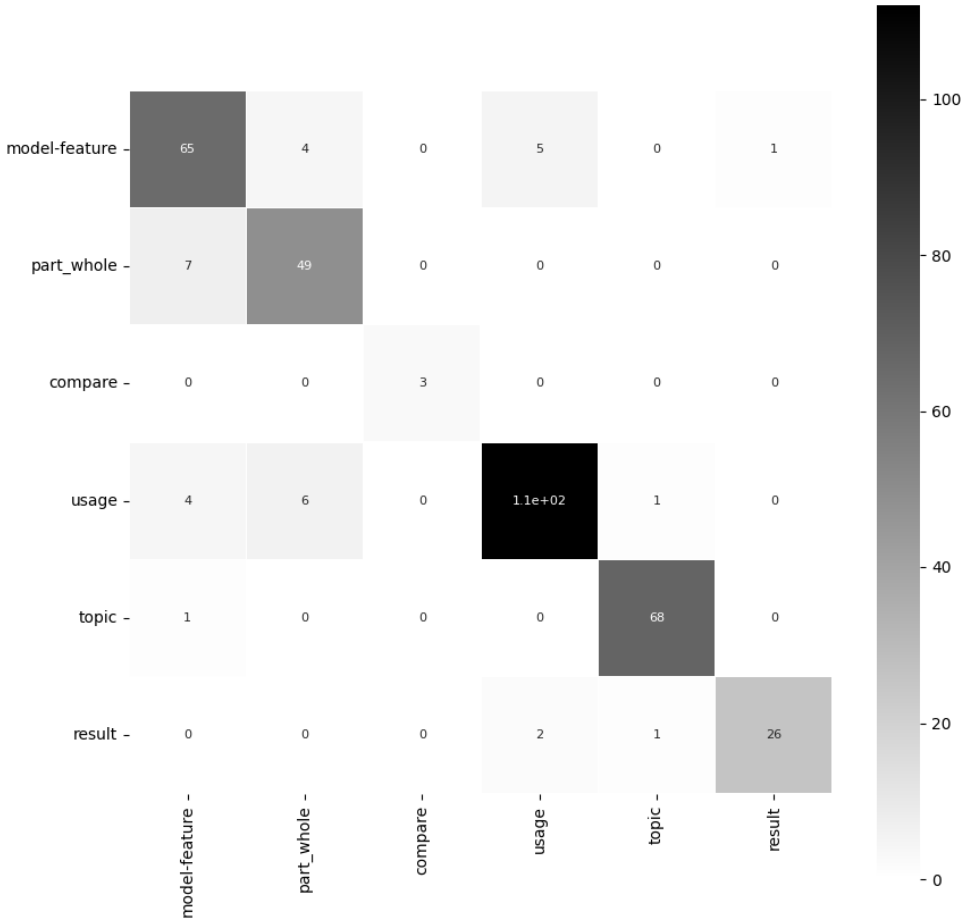

Figura 28 – Matriz de confusão para o melhor resultado do SemEval 2018 ST2.

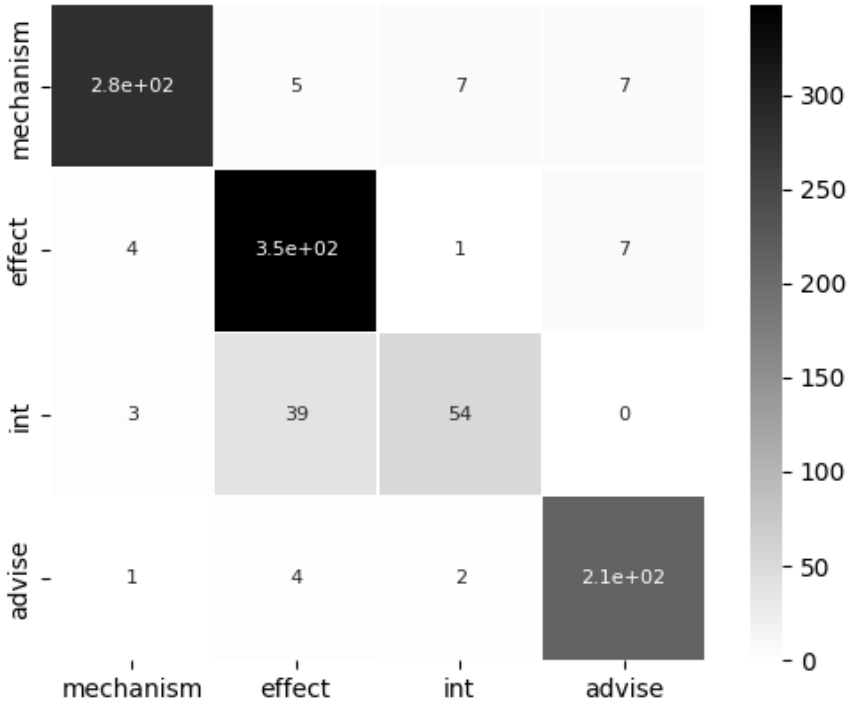

Figura 29 – Matriz de confusão para o melhor resultado do DDI 2013.

6 Conclusão

A tarefa de ER, especificamente a tarefa de CR, é muito importante no entendimento da linguagem e seu conhecimento é bastante utilizado em várias outras tarefas, tais como *machine translation*, *question-answering*, sumarização de texto, entre outras. Existem vários tipos de datasets, métodos e técnicas que são utilizadas para melhoria dos modelos de CR. Porém, a reprodutibilidade dos trabalhos em classificação de relações é difícil por conta de vários fatores como a não disponibilidade do código-fonte, ou omissão de informações importantes dos modelos como os detalhes das informações dos hiperparâmetros. Além disso, mesmo com a disponibilidade do código-fonte, alguns códigos careciam de modularidade para tornar-se extensível para o experimento de outros datasets.

O presente trabalho propôs o *DeepREF* como *framework* para tarefas de classificação de relações baseado em DL. O *DeepREF* apresenta vários módulos, tais como, módulo PLN, onde as ferramentas de PLN (*SpaCy e Stanza*) farão o processo de *tokenização* e extrairão informações das sentenças como *POS tags*; módulo de processamento de texto dos datasets, onde os datasets serão pré-processados com o tipo escolhido pelo usuário; módulo de *encoding* de sentença e *token* (*embeddings*), em que as informações extraídas dos módulos anteriores serão codificados e estarão disponíveis para o treinamento; e, por fim, o módulo de DL e otimização de hiperparâmetros, onde os *encodings* entram no processo de aprendizagem e treinamento, e são otimizados com um *framework* de otimização de hiperparâmetros (*Optuna*). É possível facilmente inserir novos datasets, *embeddings* e criar novos datasets, tipos de pré-processamento e modelos, utilizando os já preexistentes no *framework*.

Para avaliação do *framework* foi proposto um novo modelo chamado E-BEM que utiliza o BERT-EM de (SOARES et al., 2019) juntamente com *embedding* semântico e uma MLP de 3 camadas escondidas. Apesar da simplicidade do modelo, o mesmo atingiu melhores resultados comparados aos estados-da-arte nos datasets *SemEval 2018* e DDI.

No presente trabalho, os achados importantes foram que:

1. a combinação de técnicas de pré-processamento e *embeddings* alavancam os resultados dos modelos para cassificação de relações;
2. a escolha dos hiperparâmetros corretos influenciam bastante na qualidade dos modelos de classificação;

3. hiperparâmetros como a taxa de aprendizado tem mais importância na convergência dos melhores resultados dos modelos do que outros hiperparâmetros;
4. *embeddings* de contexto são muito úteis no entendimento da linguagem e ajudam os modelos de classificação a atingirem bons resultados;
5. alguns tipos de pré-processamento não são muito úteis no contexto de classificação, dependendo do domínio do dataset. Ou seja, para cada tipo de dataset existe uma técnica de pré-processamento ideal;
6. testar outras métricas de dataset pode ser interessante pois algumas métricas funcionam melhor para datasets desbalanceados;
7. o desbalanceamento do dataset influencia no resultado, ocasionando viés para a classe que possui mais exemplos no mesmo dataset.

Em suma, pré-processar o *dataset*, escolher os *embeddings* corretos e a configuração correta dos hiperparâmetros tem grande influência na qualidade dos modelos de classificação e permitem alcançar resultados próximos ou melhores do que os já existentes. O *DeepREF* propõe módulos para todas essas tarefas. Ele permite a adição de novos *datasets*, a adição de *embeddings*, assim como uma opção de facilmente utilizar modelos do BERT pré-treinados, por conta da sua modularidade.

DeepREF possui melhores resultados comparados com outros *frameworks*, como *OpenNRE* e *REflex*. Seus resultados de desempenho são também comparáveis aos modelos estados-da-arte em vários *datasets*, tais como *DDI Extraction 2013* e *SemEval 2018 Task-7*. As combinações de *embeddings* e pré-processamento proposto por *DeepREF* mostram evidências de serem efetivos comparados a muitos sistemas estados-da-arte de classificação de relações.

6.1 Limitações

O presente trabalho teve suas limitações de pesquisa que são listadas abaixo:

- falta de experimentos de validação cruzada como a autora (CHAUHAN et al., 2019) fez em seu trabalho para analisar a relevância estatística dos experimentos. Pois é possível que alterando aleatoriamente o conjunto de treino e validação, altere também o resultado no dataset de teste;
- a capacidade de memória da GPU utilizada para os experimentos. Dessa forma, não

foi possível testar algumas configurações como maiores tamanhos de *batch* pois isso extrapolaria a capacidade de memória do sistema;

- pequena quantidade de datasets avaliados. Para que houvesse uma forte evidência de que o *framework* funciona bem para qualquer dataset de qualquer domínio, era necessário que houvesse mais datasets dos mesmos domínios para serem avaliados no *framework* e, assim, embasar melhor a pesquisa com comparativos de datasets mais semelhantes;
- o *framework* só permite a classificação de relações a nível de sentença e não de relações entre sentenças. Ou seja, as sentenças nos datasets avaliados que apresentam relações entre sentenças diferentes não foram considerados durante o treinamento do modelo.

6.2 Trabalhos Futuros

Como trabalhos futuros, serão consideradas as seguintes linhas de desenvolvimento:

- a adição de mais *datasets* como o *TACRED* (ZHANG et al., 2017), que é um dataset de domínio geral e mais robusto e com mais relações do que o *SemEval 2010*, no *framework*.
- a adição de mais arquiteturas DL como GNN para avaliar dados textuais que tem relações de dependência e podem ser estruturados como grafos, como no caso do *embedding* de dependência;
- avaliar a adição de *frameworks* de otimização baseado em algoritmos genéticos pois este apresenta alguns pontos positivos, tais como o amplo espaço de busca durante a otimização e, portanto, uma maior convergência a um ótimo global;
- integrar no *framework* outros tipos de *embeddings* como *embeddings* de grafos que levam em consideração relações de dependência numa sentença, tais como o *shortest dependency-path* (SDP) e os grafos de dependência;
- adicionar mais tipos de pré-processamento como o *shortest-dependecy path* (SDP) entre duas entidades que apresenta a vantagem de obter apenas as palavras que contém informações essenciais para a classificação de relações;
- inserir processos de *text augmentation* no *framework* a fim de balancear as diferentes classes dos datasets, diminuindo o viés produzido pelo desbalanceamento e trazendo,

assim, melhorias nos resultados;

- lidar com relações *bag-level*, i.e., relações entre entidades pertencentes a diferentes sentenças para que sejam classificadas relações entre diferentes sentenças presentes nos datasets; e
- melhoria da arquitetura do *framework* para que os *embeddings* sejam mais facilmente acoplados ao *framework* podendo o usuário escolher entre *embeddings* não-treinados ou pré-treinados;
- disponibilizar o *framework* como serviço na Web para que pesquisadores possam se beneficiar dos modelos para encontrar mais rapidamente artigos ou publicações que possuam os assuntos e a relação entre eles;
- realizar testes não-funcionais no *DeepREF* para verificar melhor a usabibilidade, a escalabilidade e a performance do *framework*. Determinar também a facilidade de reuso dos módulos do *framework* para uma possível extensão das funcionalidades.

Referências

ACADEMY, D. S. **O Perceptron - Parte 1**. 2022. Disponível em: <https://www.deeplearningbook.com.br/>.

ACADEMY, S. **Framework vs library: Full comparison**. 2021. Disponível em: <https://www.interviewbit.com/blog/framework-vs-library/#:~:text=Libraries\%20provide\%20developers\%20with\%20predefined,build\%20applications\%20for\%20specific\%20platforms.>

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. Optuna: A next-generation hyperparameter optimization framework. In: **Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 2019.

ALZUBAIDI, L.; ZHANG, J.; HUMAIDI, A. J.; AL-DUJAILI, A.; DUAN, Y.; AL-SHAMMA, O.; SANTAMARíA, J.; FADHEL, M. A.; AL-AMIDIE, M.; FARHAN, L.; AL. et. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. **Journal of Big Data**, v. 8, n. 1, 2021.

ASADA, M.; MIWA, M.; SASAKI, Y. Enhancing drug-drug interaction extraction from texts by molecular structure information. In: **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Melbourne, Australia: Association for Computational Linguistics, 2018. p. 680–685. Disponível em: <https://aclanthology.org/P18-2108>.

_____. Using drug descriptions and molecular structures for drug–drug interaction extraction from literature. **Bioinformatics**, v. 37, n. 12, p. 1739–1746, 10 2020. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/btaa907>.

ASIF, N. A.; SARKER, Y.; CHAKRABORTTY, R. K.; RYAN, M. J.; AHAMED, M. H.; SAHA, D. K.; BADAL, F. R.; DAS, S. K.; ALI, M. F.; MOYEEN, S. I.; ISLAM, M. R.; TASNEEM, Z. Graph neural network: A comprehensive review on non-euclidean space. **IEEE Access**, v. 9, p. 60588–60606, 2021.

ASSIS, P. H. R. D. Supervisao a distancia em extracao de relacionamentos usando caracteristicas baseadas em hierarquia de classes em ontologias. **PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO**, 2015.

AUGENSTEIN, I.; MAYNARD, D.; CIRAVEGNA, F. Relation extraction from the web using distant supervision. In: JANOWICZ, K.; SCHLOBACH, S.; LAMBRIX, P.; HYVÖNEN, E. (Ed.). **Knowledge Engineering and Knowledge Management**. Cham: Springer International Publishing, 2014. p. 26–41. ISBN 978-3-319-13704-9.

BACH, N.; BADASKAR, S. Review of relation extraction. **Carnegie Mellon University**, 2007.

BARRIO, P.; SIMõES, G.; GALHARDAS, H.; GRAVANO, L. Reel: A relation extraction learning framework. In: **IEEE/ACM Joint Conference on Digital Libraries**. [S.l.: s.n.], 2014. p. 455–456.

BASTOS, A.; NADGERI, A.; SINGH, K.; MULANG', I. O.; SHEKARPOUR, S.; HOFFART, J. RECON: relation extraction using knowledge graph context in a graph neural network. **CoRR**, abs/2009.08694, 2020. Disponível em: <https://arxiv.org/abs/2009.08694>.

BELTAGY, I.; COHAN, A.; LO, K. Scibert: Pretrained contextualized embeddings for scientific text. **CoRR**, abs/1903.10676, 2019. Disponível em: <http://arxiv.org/abs/1903.10676>.

BERGSTRA, J.; KOMER, B.; ELIASMITH, C.; YAMINS, D.; COX, D. D. Hyperopt: a python library for model selection and hyperparameter optimization. **Computational Science &amp Discovery**, IOP Publishing, v. 8, n. 1, p. 014008, jul 2015. Disponível em: <https://doi.org/10.1088/1749-4699/8/1/014008>.

BERGSTRA, J.; YAMINS, D.; COX, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: DASGUPTA, S.; MCALLESTER, D. (Ed.). **Proceedings of the 30th International Conference on Machine Learning**. Atlanta, Georgia, USA: PMLR, 2013. (Proceedings of Machine Learning Research, 1), p. 115–123. Disponível em: <https://proceedings.mlr.press/v28/bergstra13.html>.

BODENREIDER, O. The unified medical language system (umls): integrating biomedical terminology. **Nucleic Acids Res.**, v. 32, n. Database-Issue, p. 267–270, 2004. Disponível em: <http://dblp.uni-trier.de/db/journals/nar/nar32.html#Bodenreider04>.

BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. Enriching word vectors with subword information. **CoRR**, abs/1607.04606, 2016. Disponível em: <http://arxiv.org/abs/1607.04606>.

BOLLACKER, K.; COOK, R.; TUFTS, P. Freebase: A shared database of structured general human knowledge. In: **Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2**. [S.l.]: AAAI Press, 2007. (AAAI'07), p. 1962–1963. ISBN 9781577353232.

BOUKKOURI, H. E.; FERRET, O.; LAVERGNE, T.; NOJI, H.; ZWEIGENBAUM, P.; TSUJII, J. Characterbert: Reconciling elmo and BERT for word-level open-vocabulary representations from characters. **CoRR**, abs/2010.10392, 2020. Disponível em: <https://arxiv.org/abs/2010.10392>.

CAI, R.; ZHANG, X.; WANG, H. Bidirectional recurrent convolutional neural network for relation classification. In: **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Berlin, Germany: Association for Computational Linguistics, 2016. p. 756–765. Disponível em: <https://aclanthology.org/P16-1072>.

CHAUHAN, G.; MCDERMOTT, M. B.; SZOLOVITS, P. REflex: Flexible framework for relation extraction in multiple domains. In: **Proceedings of the 18th BioNLP Workshop and Shared Task**. Florence, Italy: Association for Computational Linguistics, 2019. p. 30–47. Disponível em: <https://aclanthology.org/W19-5004>.

CHUAN, L.; QUANYUAN, F. The standard particle swarm optimization algorithm convergence analysis and parameter selection. In: **Third International Conference on Natural Computation (ICNC 2007)**. [S.l.: s.n.], 2007. v. 3, p. 823–826.

COHEN, A. D. N.; ROSENMAN, S.; GOLDBERG, Y. Relation extraction as two-way span-prediction. **CoRR**, abs/2010.04829, 2020. Disponível em: <https://arxiv.org/abs/2010.04829>.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: <https://aclanthology.org/N19-1423>.

EDUCATION, I. C. **What are neural networks?** 2020. Disponível em: <https://www.ibm.com/cloud/learn/neural-networks#:~:text=Neural\%20networks\%2C\%20also\%20known\%20as,neurons\%20signal\%20to\%20one\%20another.>

ELGELDAWI, E.; SAYED, A.; GALAL, A. R.; ZAKI, A. M. Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis. **Informatics**, v. 8, n. 4, 2021. ISSN 2227-9709. Disponível em: <https://www.mdpi.com/2227-9709/8/4/79>.

FELLBAUM, C. (Ed.). **WordNet: an electronic lexical database**. [S.l.]: MIT Press, 1998.

FIRTH, J. R. A synopsis of linguistic theory 1930-55. The Philological Society, Oxford, v. 1952-59, p. 1–32, 1957.

GÁBOR, K.; BUSCALDI, D.; SCHUMANN, A.-K.; QASEMIZADEH, B.; ZARGAYOUNA, H.; CHARNOIS, T. SemEval-2018 task 7: Semantic relation extraction and classification in scientific papers. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 679–688. Disponível em: <https://aclanthology.org/S18-1111>.

GAIZAUSKAS, R.; WILKS, Y. Information extraction: Beyond document retrieval. In: **International Journal of Computational Linguistics & Chinese Language Processing, Volume 3, Number 2, August 1998**. [s.n.], 1998. p. 17–60. Disponível em: <https://aclanthology.org/O98-4002>.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design patterns: Elements of reusable object-oriented software**. [S.l.]: Addison-Wesley professional computing series, 2021.

GERS, F.; SCHMIDHUBER, J. Recurrent nets that time and count. In: **Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium**. [S.l.: s.n.], 2000. v. 3, p. 189–194 vol.3.

GOLOVIN, D.; SOLNIK, B.; MOITRA, S.; KOCHANSKI, G.; KARRO, J. E.; SCULLEY, D. (Ed.). **Google Vizier: A Service for Black-Box Optimization**. [s.n.], 2017. 1487–1495 p. Disponível em: <http://www.kdd.org/kdd2017/papers/view/google-vizier-a-service-for-black-box-optimization>.

GUO, X.; ZHANG, H.; YANG, H.; XU, L.; YE, Z. A single attention-based combination of cnn and rnn for relation classification. **IEEE Access**, v. 7, p. 12467–12475, 2019.

GUO, Z.; ZHANG, Y.; LU, W. Attention guided graph convolutional networks for relation extraction. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Florence, Italy: Association for Computational Linguistics, 2019. p. 241–251. Disponível em: <https://aclanthology.org/P19-1024>.

HAN, X.; GAO, T.; YAO, Y.; YE, D.; LIU, Z.; SUN, M. OpenNRE: An open and extensible toolkit for neural relation extraction. In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations**. Hong Kong, China: Association for Computational Linguistics, 2019. p. 169–174. Disponível em: <https://aclanthology.org/D19-3029>.

HAN, X.; ZHU, H.; YU, P.; WANG, Z.; YAO, Y.; LIU, Z.; SUN, M. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing**. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 4803–4809. Disponível em: <https://aclanthology.org/D18-1514>.

HENDRICKX, I.; KIM, S. N.; KOZAREVA, Z.; NAKOV, P.; SÉAGHDHA, D. Ó.; PADÓ, S.; PENNACCHIOTTI, M.; ROMANO, L.; SZPAKOWICZ, S. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In: **Proceedings of the 5th International Workshop on Semantic Evaluation**. Uppsala, Sweden: Association for Computational Linguistics, 2010. p. 33–38. Disponível em: <https://aclanthology.org/S10-1006>.

HERRERO-ZAZO, M.; SEGURA-BEDMAR, I.; MARTíNEZ, P.; DECLERCK, T. The ddi corpus: An annotated corpus with pharmacological substances and drug–drug interactions. **Journal of Biomedical Informatics**, v. 46, n. 5, p. 914–920, 2013. ISSN 1532-0464. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1532046413001123>.

HETTINGER, L.; DALLMANN, A.; ZEHE, A.; NIEBLER, T.; HOTHO, A. ClaiRE at SemEval-2018 task 7: Classification of relations using embeddings. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 836–841. Disponível em: <https://aclanthology.org/S18-1134>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Sequential model-based optimization for general algorithm configuration. In: COELLO, C. A. C. (Ed.). **Learning and Intelligent Optimization**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 507–523. ISBN 978-3-642-25566-3.

JAMIESON, K. G.; TALWALKAR, A. Non-stochastic best arm identification and hyperparameter optimization. **CoRR**, abs/1502.07943, 2015. Disponível em: <http://arxiv.org/abs/1502.07943>.

JIN, D.; DERNONCOURT, F.; SERGEEVA, E.; MCDERMOTT, M.; CHAUHAN, G. MIT-MEDG at SemEval-2018 task 7: Semantic relation classification via convolution

neural network. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 798–804. Disponível em: <https://aclanthology.org/S18-1127>.

JIN, W.; CAI, Y.; KAZMAN, R.; ZHENG, Q.; CUI, D.; LIU, T. Enre: A tool framework for extensible entity relation extraction. In: **2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)**. [S.l.: s.n.], 2019. p. 67–70.

KAMATH, U.; LIU, J.; WHITAKER, J. **Deep Learning for NLP and Speech Recognition**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2019. ISBN 3030145956.

KAUFMANN, E.; GARIVIER, A. Learning the distribution with largest mean: two bandit frameworks. **CoRR**, abs/1702.00001, 2017. Disponível em: <http://arxiv.org/abs/1702.00001>.

KOCH, P.; GOLOVIDOV, O.; GARDNER, S.; WUJEK, B.; GRIFFIN, J.; XU, Y. Autotune: A derivative-free optimization framework for hyperparameter tuning. In: **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining**. New York, NY, USA: Association for Computing Machinery, 2018. (KDD '18), p. 443–452. ISBN 9781450355520. Disponível em: <https://doi.org/10.1145/3219819.3219837>.

KOEHRSEN, W. **A conceptual explanation of Bayesian hyperparameter optimization for Machine Learning**. Towards Data Science, 2018. Disponível em: <https://bit.ly/2lYVvXU>.

KRALLINGER, M.; RABAL, O.; AKHONDI, S. A.; PÉREZ, M. P.; SANTAMARÍA, J.; RODRÍGUEZ, G. P.; TSATSARONIS, G.; INTXAURRONDO, A.; LOPEZ, J. A.; NANDAL, U. K.; BUEL, E. M. van; CHANDRASEKHAR, A.; RODENBURG, M.; LÆGREID, A.; DOORNENBAL, M. A.; OYARZÁBAL, J.; LOURENçO, A.; VALENCIA, A. Overview of the biocreative vi chemical-protein interaction track. In: . [S.l.: s.n.], 2017.

KUMAR, S. A survey of deep learning methods for relation extraction. **CoRR**, abs/1705.03645, 2017. Disponível em: <http://arxiv.org/abs/1705.03645>.

LEE, J.; YOON, W.; KIM, S.; KIM, D.; KIM, S.; SO, C. H.; KANG, J. Biobert: a pre-trained biomedical language representation model for biomedical text mining. **Bioinformatics**, Oxford University Press (OUP), Sep 2019. ISSN 1460-2059. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btz682>.

LEUNG, K. **Micro, Macro amp; weighted averages of F1 score, clearly explained**. Towards Data Science, 2022. Disponível em: <https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f>.

LI, C.; TIAN, Y. Downstream model design of pre-trained language model for relation extraction task. **CoRR**, abs/2004.03786, 2020. Disponível em: <https://arxiv.org/abs/2004.03786>.

LIAW, R.; LIANG, E.; NISHIHARA, R.; MORITZ, P.; GONZALEZ, J. E.; STOICA, I. Tune: A research platform for distributed model selection and training. **CoRR**, abs/1807.05118, 2018. Disponível em: <http://arxiv.org/abs/1807.05118>.

LIMA, R.; FREITAS, F. L. G. d. Ontoilper: An ontology- and inductive logic programming-based system to extract entities and relations from text. **Universidade Federal de Pernambuco**, p. 223–255, Jan 2014.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized BERT pretraining approach. **CoRR**, abs/1907.11692, 2019. Disponível em: <http://arxiv.org/abs/1907.11692>.

LOBO, F. G.; GOLDBERG, D. E.; PELIKAN, M. Time complexity of genetic algorithms on exponentially scaled problems. In: **Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. (GECCO'00), p. 151–158. ISBN 1558607080.

LOPER, E.; BIRD, S. Nltk: The natural language toolkit. **CoRR**, cs.CL/0205028, 2002. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr0205.html#cs-CL-0205028>.

LUAN, Y.; OSTENDORF, M.; HAJISHIRZI, H. The UWNLP system at SemEval-2018 task 7: Neural relation extraction model with selectively incorporated concept embeddings. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 788–792. Disponível em: <https://aclanthology.org/S18-1125>.

MACAVANEY, S.; SOLDAINI, L.; COHAN, A.; GOHARIAN, N. GU IRLAB at SemEval-2018 task 7: Tree-LSTMs for scientific relation classification. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 831–835. Disponível em: <https://aclanthology.org/S18-1133>.

MARCUS, M. P.; SANTORINI, B.; MARCINKIEWICZ, M. A. Building a large annotated corpus of English: The Penn Treebank. **Computational Linguistics**, MIT Press, Cambridge, MA, v. 19, n. 2, p. 313–330, 1993. Disponível em: <https://aclanthology.org/J93-2004>.

MCDONALD, R.; NIVRE, J.; QUIRMBACH-BRUNDAGE, Y.; GOLDBERG, Y.; DAS, D.; GANCHEV, K.; HALL, K.; PETROV, S.; ZHANG, H.; TÄCKSTRÖM, O.; BEDINI, C.; CASTELLÓ, N. B.; LEE, J. Universal Dependency annotation for multilingual parsing. In: **Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Sofia, Bulgaria: Association for Computational Linguistics, 2013. p. 92–97. Disponível em: <https://aclanthology.org/P13-2017>.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. In: **1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings**. [s.n.], 2013. Disponível em: <http://arxiv.org/abs/1301.3781>.

MUZAFFAR, A.; AZAM, F.; QAMAR, U. A relation extraction framework for biomedical text using hybrid feature set. **Computational and Mathematical Methods in Medicine**, v. 2015, 08 2015.

MWENDI, E. Software frameworks, architectural and design patterns. **Journal of Software Engineering and Applications**, v. 07, p. 670–678, 01 2014.

NASCIMENTO, I.; LIMA, R.; CHIFU, A.-G.; ESPINASSE, B.; FOURNIER, S. Deepref: A framework for optimized deep learning-based relation classification. In: **Proceedings of the Language Resources and Evaluation Conference**. Marseille, France: European Language Resources Association, 2022. p. 4513–4522. Disponível em: <https://aclanthology.org/2022.lrec-1.480>.

NASTASE, V.; NAKOV, P.; SAGHDHA, D.; SZPAKOWICZ, S. **Semantic Relations Between Nominals**. [S.l.]: Morgan & Claypool Publishers, 2013. ISBN 1608459799.

NGUYEN, T. H.; GRISHMAN, R. Relation extraction: Perspective from convolutional neural networks. In: **Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing**. Denver, Colorado: Association for Computational Linguistics, 2015. p. 39–48. Disponível em: <https://aclanthology.org/W15-1506>.

NOORALAHZADEH, F.; ØVRELID, L.; LØNNING, J. T. SIRIUS-LTG-UiO at SemEval-2018 task 7: Convolutional neural networks with shortest dependency paths for semantic relation extraction and classification in scientific papers. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 805–810. Disponível em: <https://aclanthology.org/S18-1128>.

PAWAR, S.; PALSHIKAR, G. K.; BHATTACHARYYA, P. Relation extraction : A survey. **CoRR**, abs/1712.05191, 2017. Disponível em: <http://arxiv.org/abs/1712.05191>.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Empirical Methods in Natural Language Processing (EMNLP)**. [s.n.], 2014. p. 1532–1543. Disponível em: <http://www.aclweb.org/anthology/D14-1162>.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. **CoRR**, abs/1802.05365, 2018. Disponível em: <http://arxiv.org/abs/1802.05365>.

PHAN, L. N.; ANIBAL, J. T.; TRAN, H.; CHANANA, S.; BAHADROGLU, E.; PELTEKIAN, A.; ALTAN-BONNET, G. Scifive: a text-to-text transformer model for biomedical literature. **CoRR**, abs/2106.03598, 2021. Disponível em: <https://arxiv.org/abs/2106.03598>.

PILEHVAR, M. T.; CAMACHO-COLLADOS, J. Embeddings in natural language processing: Theory and advances in vector representations of meaning. **Synthesis Lectures on Human Language Technologies**, v. 13, n. 4, p. 1–175, 2020. Disponível em: <https://doi.org/10.2200/S01057ED1V01Y202009HLT047>.

PRATAP, B.; SHANK, D.; OSITELU, O.; GALBRAITH, B. Talla at SemEval-2018 task 7: Hybrid loss optimization for relation classification using convolutional neural networks. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 863–867. Disponível em: <https://aclanthology.org/S18-1139>.

QIN, P.; XU, W.; GUO, J. Designing an adaptive attention mechanism for relation classification. In: **2017 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2017. p. 4356–4362.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. Improving language understanding by generative pre-training. 2018.

RAUF, H. T.; SHOAIB, U.; LALI, M. I.; ALHAISONI, M.; IRFAN, M. N.; KHAN, M. A. Particle swarm optimization with probability sequence for global optimization. **IEEE Access**, v. 8, p. 110535–110549, 2020.

REN, F.; ZHOU, D.; LIU, Z.; LI, Y.; ZHAO, R.; LIU, Y.; LIANG, X. Neural relation classification with text descriptions. In: **Proceedings of the 27th International Conference on Computational Linguistics**. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018. p. 1167–1177. Disponível em: <https://aclanthology.org/C18-1100>.

RIEHLE, D. Framework design: A role modeling approach. **Softwaretechnik-Trends**, v. 20, 01 2000.

ROTSZTEJN, J.; HOLLENSTEIN, N.; ZHANG, C. ETH-DS3Lab at SemEval-2018 task 7: Effectively combining recurrent and convolutional neural networks for relation classification and extraction. In: **Proceedings of The 12th International Workshop on Semantic Evaluation**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 689–696. Disponível em: <https://aclanthology.org/S18-1112>.

SAHU, S. K.; THOMAS, D.; CHIU, B.; SENGUPTA, N.; MAHDY, M. Relation extraction with self-determined graph convolutional network. **CoRR**, abs/2008.00441, 2020. Disponível em: <https://arxiv.org/abs/2008.00441>.

SANG, E. F. T. K.; MEULDER, F. D. 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.n.], 2003. p. 142–147. Disponível em: <https://aclanthology.org/W03-0419>.

SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. **CoRR**, abs/1910.01108, 2019. Disponível em: <http://arxiv.org/abs/1910.01108>.

SANTOS, C. dos; XIANG, B.; ZHOU, B. Classifying relations by ranking with convolutional neural networks. In: **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Beijing, China: Association for Computational Linguistics, 2015. p. 626–634. Disponível em: <https://aclanthology.org/P15-1061>.

SCARSELLI, F.; GORI, M.; TSOI, A. C.; HAGENBUCHNER, M.; MONFARDINI, G. The graph neural network model. **IEEE Transactions on Neural Networks**, v. 20, n. 1, p. 61–80, 2009.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. **Practical Bayesian Optimization of Machine Learning Algorithms**. arXiv, 2012. Disponível em: <https://arxiv.org/abs/1206.2944>.

SOARES, L. B.; FITZGERALD, N.; LING, J.; KWIATKOWSKI, T. Matching the blanks: Distributional similarity for relation learning. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Florence, Italy: Association for Computational Linguistics, 2019. p. 2895–2905. Disponível em: <https://aclanthology.org/P19-1279>.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: Pretrained bert models for brazilian portuguese. In: CERRI, R.; PRATI, R. C. (Ed.). **Intelligent Systems**. Cham: Springer International Publishing, 2020. p. 403–417. ISBN 978-3-030-61377-8.

STROBL, C.; BOULESTEIX, A.-L.; KNEIB, T.; AUGUSTIN, T.; ZEILEIS, A. Conditional variable importance for random forests. **BMC Bioinformatics**, BioMed Central Ltd., v. 9, n. 307, p. 1–11, July 2008. CS defined the research question, suggested the conditional variable importance, set up and performed the simulation experiments and drafted the manuscript. A-LB analyzed the peptide-binding data. TK, TA and AZ contributed to the theoretical understanding and presentation of the problem. All authors contributed to and approved the final version of the manuscript. Disponível em: <https://epub.wu.ac.at/4987/>.

TAKANOBU, R.; ZHANG, T.; LIU, J.; HUANG, M. A hierarchical framework for relation extraction with reinforcement learning. In: **Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence**. AAAI Press, 2019. (AAAI'19/IAAI'19/EAAI'19). ISBN 978-1-57735-809-1. Disponível em: <https://doi.org/10.1609/aaai.v33i01.33017072>.

TAO, Q.; LUO, X.; WANG, H. Enhancing relation extraction using syntactic indicators and sentential contexts. **CoRR**, abs/1912.01858, 2019. Disponível em: <http://arxiv.org/abs/1912.01858>.

UZUNER, ; SOUTH, B. R.; SHEN, S.; DUVALL, S. L. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. **Journal of the American Medical Informatics Association**, v. 18, n. 5, p. 552–556, 2011.

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

VRANDECIC, D. Wikidata: A new platform for collaborative data collection. In: **Proceedings of the 21st International Conference on World Wide Web**. New York, NY, USA: Association for Computing Machinery, 2012. (WWW '12 Companion), p. 1063–1064. ISBN 9781450312301. Disponível em: <https://doi.org/10.1145/2187980.2188242>.

WANG, H.; QIN, K.; LU, G.; LUO, G.; LIU, G. Direction-sensitive relation extraction using bi-sdp attention model. **Knowledge-Based Systems**, v. 198, p. 105928, 2020. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705120302628>.

WANG, H.; QIN, K.; ZAKARI, R. Y.; LIU, G.; LU, G. Deep neural network based relation extraction: An overview. **CoRR**, abs/2101.01907, 2021. Disponível em: <https://arxiv.org/abs/2101.01907>.

WANG, L.; CAO, Z.; MELO, G. de; LIU, Z. Relation classification via multi-level attention CNNs. In: **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Berlin, Germany: Association for Computational Linguistics, 2016. p. 1298–1307. Disponível em: <https://aclanthology.org/P16-1123>.

WU, F.; ZHANG, T.; JR., A. H. S.; FIFTY, C.; YU, T.; WEINBERGER, K. Q. Simplifying graph convolutional networks. **CoRR**, abs/1902.07153, 2019. Disponível em: <http://arxiv.org/abs/1902.07153>.

XIAO, M.; LIU, C. Semantic relation classification via hierarchical recurrent neural network with attention. In: **Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers**. Osaka, Japan: The COLING 2016 Organizing Committee, 2016. p. 1254–1263. Disponível em: <https://aclanthology.org/C16-1119>.

XU, K.; FENG, Y.; HUANG, S.; ZHAO, D. Semantic relation classification via convolutional neural networks with simple negative sampling. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 536–540. Disponível em: <https://aclanthology.org/D15-1062>.

XU, Y.; JIA, R.; MOU, L.; LI, G.; CHEN, Y.; LU, Y.; JIN, Z. Improved relation classification by deep recurrent neural networks with data augmentation. In: **Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers**. Osaka, Japan: The COLING 2016 Organizing Committee, 2016. p. 1461–1470. Disponível em: <https://aclanthology.org/C16-1138>.

YAN, X.-H.; HE, F.-Z.; CHEN, Y.-L. A novel hardware/software partitioning method based on position disturbed particle swarm optimization with invasive weed optimization. **Journal of Computer Science and Technology**, v. 32, n. 2, p. 340–355, 2017.

YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. **Neurocomputing**, v. 415, p. 295–316, 2020. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231220311693>.

YANG, Z.; DAI, Z.; YANG, Y.; CARBONELL, J. G.; SALAKHUTDINOV, R.; LE, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. **CoRR**, abs/1906.08237, 2019. Disponível em: <http://arxiv.org/abs/1906.08237>.

YU, T.; ZHU, H. Hyper-parameter optimization: A review of algorithms and applications. **CoRR**, abs/2003.05689, 2020. Disponível em: <https://arxiv.org/abs/2003.05689>.

ZENG, D.; LIU, K.; CHEN, Y.; ZHAO, J. Distant supervision for relation extraction via piecewise convolutional neural networks. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 1753–1762. Disponível em: <https://aclanthology.org/D15-1203>.

ZENG, D.; LIU, K.; LAI, S.; ZHOU, G.; ZHAO, J. Relation classification via convolutional deep neural network. In: **Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers**. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, 2014. p. 2335–2344. Disponível em: <https://aclanthology.org/C14-1220>.

ZHANG, C.; CUI, C.; GAO, S.; NIE, X.; XU, W.; YANG, L.; XI, X.; YIN, Y. Multi-gram cnn-based self-attention model for relation classification. **IEEE Access**, v. 7, p. 5343–5357, 2019.

ZHANG, D.; WANG, D. Relation classification via recurrent neural network. **CoRR**, abs/1508.01006, 2015. Disponível em: <http://arxiv.org/abs/1508.01006>.

ZHANG, L.; XIANG, F. Relation classification via bilstm-cnn. In: TAN, Y.; SHI, Y.; TANG, Q. (Ed.). **Data Mining and Big Data**. Cham: Springer International Publishing, 2018. p. 373–382. ISBN 978-3-319-93803-5.

ZHANG, Y.; QI, P.; MANNING, C. D. Graph convolution over pruned dependency trees improves relation extraction. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing**. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 2205–2215. Disponível em: <https://aclanthology.org/D18-1244>.

ZHANG, Y.; ZHONG, V.; CHEN, D.; ANGELI, G.; MANNING, C. D. Position-aware attention and supervised data improve slot filling. In: **Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing**. Copenhagen, Denmark: Association for Computational Linguistics, 2017. p. 35–45. Disponível em: <https://aclanthology.org/D17-1004>.

ZHAO, K.; XU, H.; CHENG, Y.; LI, X.; GAO, K. Representation iterative fusion based on heterogeneous graph neural network for joint entity and relation extraction. **Knowledge-Based Systems**, v. 219, p. 106888, 2021. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705121001519>.

ZHENG, S.; XU, J.; ZHOU, P.; BAO, H.; QI, Z.; XU, B. A neural network framework for relation extraction. **Know.-Based Syst.**, Elsevier Science Publishers B. V., NLD, v. 114, n. C, p. 12–23, dec 2016. ISSN 0950-7051. Disponível em: <https://doi.org/10.1016/j.knosys.2016.09.019>.

ZHOU, P.; SHI, W.; TIAN, J.; QI, Z.; LI, B.; HAO, H.; XU, B. Attention-based bidirectional long short-term memory networks for relation classification. In: **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Berlin, Germany: Association for Computational Linguistics, 2016. p. 207–212. Disponível em: <https://aclanthology.org/P16-2034>.

ZHU, H.; LIN, Y.; LIU, Z.; FU, J.; CHUA, T.-S.; SUN, M. Graph neural networks with generated parameters for relation extraction. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Florence, Italy: Association for Computational Linguistics, 2019. p. 1331–1339. Disponível em: <https://aclanthology.org/P19-1128>.

APÊNDICES

APÊNDICE A – Estatísticas das quantidades de sentenças por tamanho para cada dataset avaliado no *DeepREF* sem pré-processamento

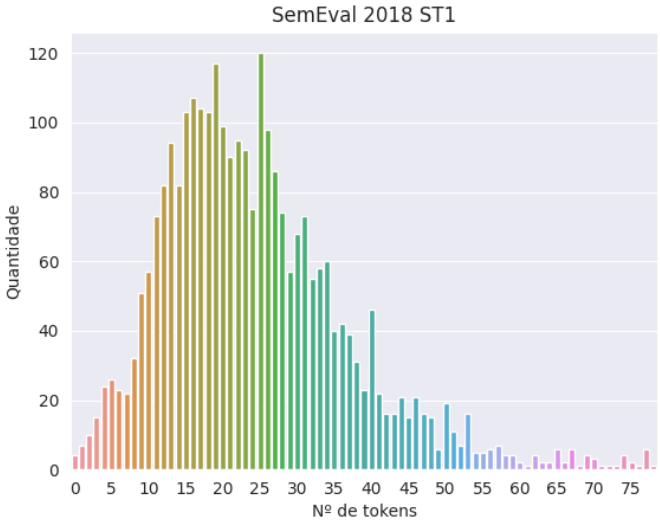

Figura 30 – Estatística da quantidade de sentenças por tamanho para o dataset SemEval 2018 ST1.

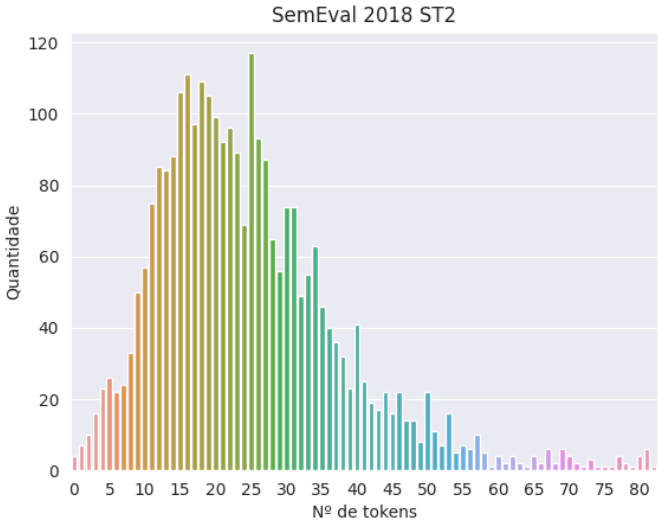

Figura 31 – Estatística da quantidade de sentenças por tamanho para o dataset SemEval 2018 ST2.

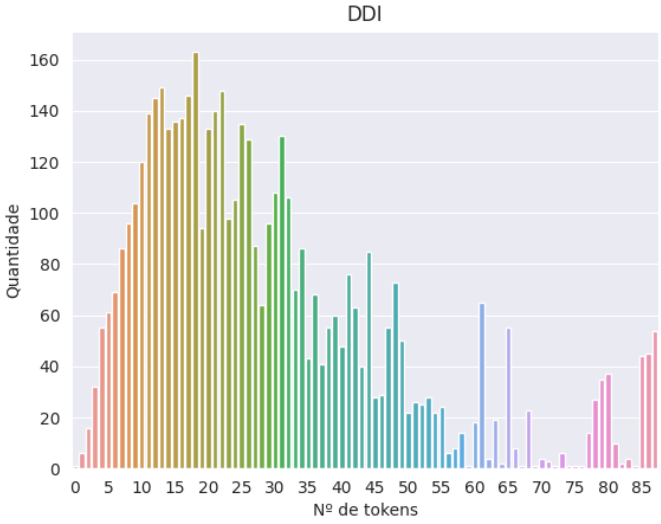

Figura 32 – Estatística da quantidade de sentenças por tamanho para o dataset DDI.