PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EXTRAÇÃO DE RELACIONAMENTOS EM TEXTOS DE LÍNGUA PORTUGUESA UTILIZANDO REDES NEURAIS RECORRENTES

BRUNO DORSCHEIDT BRANDELLI

Trabalho de Conclusão II apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Renata Vieira

Porto Alegre
2019

EXTRAÇÃO DE RELACIONAMENTOS EM TEXTOS DE LÍNGUA PORTUGUESA UTILIZANDO REDES NEURAIS RECORRENTES

RESUMO

O Processamento de Linguagem Natural é cada vez mais necessário nos dias atuais, visto que a quantidade de dados desestruturados está cada vez maior e espalhada por todo o mundo, fazendo com que a necessidade de torná-los estruturados e imbuir significado para eles também. Para isso existe a Extração de Informação, que consegue atingir este objetivo, mas como existem inúmeras línguas no mundo, são necessárias diferentes abordagens e sistemas para extrair estas informações significativas. Sendo assim, o objetivo principal deste trabalho é o desenvolvimento de um sistema utilizando o modelo de Redes Neurais Recorrentes, que seja capaz de realizar tarefas de Extração de Informação como a Extração de Relacionamentos em textos de língua portuguesa.

Palavras-Chave: Processamento de Linguagem Natural, Extração de Informação, Entidades Nomeadas, Extração de Relacionamentos, Redes Neurais Artificiais, Redes Neurais Recorrentes.

RELATIONSHIP EXTRACTION IN PORTUGUESE TEXTS USING ARTIFICIAL NEURAL NETWORKS

ABSTRACT

Nowadays Natural Language Processing is even more needed, due the amount of unestructured data spread around the world, raising the necessity to make them structured and put significance into them. Such thing is possible through Information Extraction, that is capable to reach this objective, but, there are a large number of languages around the world, making necessary to use different approachs and systems to be able to extract valuable data. Thus, the main goal of this work is develop a system based on Recurrent Neural Networks model, capable of accomplish Information Extraction tasks, such as Relationship Extraction in portuguese texts.

Keywords: Natural Language Processing, Information Extraction, Named Entities, Relationship Extraction, Artificial Neural Networks, Recurrent Neural Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CRF – Conditional Random Fields

EI – Extração de Informação

ER – Extração de Relacionamento

IA – Inteligência Artificial

IBERLEF – Iberian Languages Evaluation Forum

LSTM – Long Short-Term Memory

MEMM – Maximum-Entropy Markov Model

MIT – Massachusetts Institute of Technology

PLN – Processamento de Linguagem Natural

POS – Parts-of-Speech

RCC – Relacionamentos Completamente Corretos

REN – Reconhecimento de Entidades Nomeadas

RI – Relacionamentos Identificados

RNA – Redes Neurais Artificiais

RNC – Redes Neurais Convolucionais

RNR – Redes Neurais Recorrentes

RPC – Relacionamentos Parcialmente Corretos

RPC – Relacionamentos Parcialmente Corretos Absolutos

STM – Short-Term Memory

TCC – Trabalho de Conclusão de Curso

TR – Total de Relacionamentos

TSV – Tab-separated values

LISTA DE ABREVIATURAS

LOC – Local

ORG – Organização

PER – Pessoa

VEH – Veículo

SUMÁRIO

1. INTRODUÇÃO

Hoje em dia temos um número muito grande de informações em forma de texto, e dentro destes textos, muitas informações que podem ser úteis, para pessoas, empresas e até para segurança. Dentro da área de Inteligência Artificial (IA), temos a subárea de Processamento de Linguagem Natural (PLN), focada em estudar a forma com que computadores ou sistemas conseguem receber dados relativos à linguagem, como textos ou falas, e processar estes dados de forma a torná-los significativos do ponto de vista deste sistema.

Dentre as inúmeras tarefas de alto nível presentes no PLN, temos a Extração de Informação (EI), que transforma as informações desestruturadas presentes em um texto de entrada, em dados estruturados [17] que podem ser lidos, analisados e processados por sistemas. Estas entradas de textos podem conter certas informações que são relevantes para determinados indivíduos e irrelevantes para outros, e ao mesmo tempo ter dados que o primeiro individuo descartaria, mas são de suma importância para o segundo.

Para filtrar somente os dados que serão significativos, a entrada é analisada utilizando alguns parâmetros, e buscando por determinadas características. Onde podemos buscar por nomes de pessoas, objetos, localidades, eventos, relacionamentos entre os itens citados anteriormente, além de inúmeras outras coisas.

1.1 Caracterização do Problema

Visto o avanço do tamanho da massa de dados que vem a ser processado e analisado, a busca por formas mais sofisticadas e robustas de realizar a extração de informações também evolui [22]. A linguagem que possui um maior espectro de estudos é o inglês, onde é possível encontrar um maior número de textos anotados, ontologia de domínio, ferramentas e técnicas.

A língua portuguesa por outro lado, tem recursos mais escassos [4] para a realização desta tarefa, onde algumas competições tem sido a maior fonte de trabalhos desenvolvidos em algumas das tarefas presentes em EI. O HAREM *evaluation contest* [21], impulsionou as áreas de Reconhecimento de Entidades Nomeadas (REN) e Extração de Relacionamento (ER), e neste ano de 2019 teremos o *Iberian Languages Evaluation Forum* (IberLEF) [3] que propõem o desenvolvimento de sistemas para processamento e entendimento de texto para línguas ibéricas.

Sendo assim, este trabalho irá explorar o desenvolvimento de um sistema utilizando o modelo de Redes Neurais Recorrentes (RNR), como forma de abordar a tarefa de ER em textos de língua portuguesa.

1.2 Organização do Trabalho

Este documento está dividido em 10 capítulos, visando uma fácil organização dos conteúdos, apresentando-os da seguinte forma: no capítulo 2 são determinados os objetivos que são almejados neste trabalho, tanto de escopo geral como específicos. No capítulo 3 são introduzidos os conceitos teóricos básicos necessários para o desenvolvimento do trabalho. No capítulo 4 são citados alguns trabalhos relacionados a tarefa, de onde serão observadas as abordagens utilizadas e resultados obtidos. No capítulo 5 os recursos necessários para o desenvolvimento do trabalho serão detalhados, sendo eles, a linguagem de programação, ferramentas disponíveis, além do *hardware*. O capítulo 6 traz maiores detalhes da tarefa do *IberLEF* que será realizada neste trabalho. Tendo uma melhor noção do que dever ser feito, o capítulo 7 mostra as etapas de pré-processamento que os dados devem ser submetidos, para que a tarefa possa ser realizada da forma mais satisfatória possível. Com todos os recursos e dados necessários já detalhados, o capítulo 8 ilustra todas as etapas necessárias para criação do modelo final que será utilizado para a realização da tarefa proposta. Após o desenvolvimento do modelo, o capítulo 9 busca consolidar os resultados obtidos pelo modelo desenvolvido, onde será possível realizar a analise deles, e comparar com outros sistemas enviados ao *IberLEF* 2019. Finalizando, temos o capitulo 10, que é feita a conclusão deste trabalho, expondo as considerações finais, e sinalizando melhorias futuras para o sistema.

2. OBJETIVOS

2.1 Objetivos Gerais

O objetivo geral deste trabalho de conclusão é o desenvolvimento de um sistema que realize tarefas de EI, em textos que tenham a língua portuguesa como base. Onde dentre as sub-tarefas, a ER será o foco principal deste sistema. Para desenvolvimento de tal sistema serão utilizadas Redes Neurais Artificiais (RNA) e suas variações. No final, os resultados serão comparados com o trabalhos existentes na área, e com mais sistemas enviados ao *IberLEF* 2019, que poderão fornecer uma comparação melhor, visto que serão utilizados os mesmos *datasets*, além de possuir o mesmo objetivo.

2.2 Objetivos Específicos

Durante o desenvolvimento do trabalho, teremos alguns objetivos específicos principais, que são eles:

1. Aprofundar o conhecimento em PLN.
2. Estudar as abordagens e resultados obtidos nos estudos recentes de PLN, principalmente em ER.
3. Familiarizar-se com as tarefas relacionadas a PLN do *IberLEF* 2019, assim como com o *dataset* utilizado.
4. Estudar linguagens de programação e bibliotecas que estão em destaque na área de PLN.
5. Definir linguagem e modelo que serão utilizados para desenvolver o sistema para realizar a tarefa de ER.
6. Treinar e testar o sistema.
7. Analisar os resultados obtidos com o sistema.

3. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo criar um embasamento teórico na área de PLN, visto que o objetivo geral deste trabalho está relacionado a tarefas desta área de estudo, assim como alguns conceitos que servem de base para tal área, como Aprendizado de Máquina e RNA. Além de trazer conceitos básicos das tarefas de EI, REN e ER.

3.1 Aprendizado de Máquina

Aprendizado de Máquina pode ser definido como uma subárea da Ciência da Computação, que tem como objetivo, dado um problema prático, por meio de coleta de um *dataset* pertinente ao domínio do problema e algoritmicamente criar um modelo estatístico baseado no *dataset*, chegando no final a resolução do problema utilizando o modelo desenvolvido [6]. Uma vasta gama de aplicações utiliza algoritmos de Aprendizado de Máquina, sendo elas, filtro de *e-mail*, visão computacional, detecção de fraudes, classificação de textos, entre outros.

Abaixo temos os principais tipos de aprendizagem que podem contribuir no desenvolvimento deste trabalho, junto de suas principais características, conforme descrito por Burkov [6]:

- **Supervisionada:** Utiliza um *dataset* anotado para criar o modelo, e busca classificar novas entradas utilizando as características aprendidas do *dataset*.
- **Não-Supervisionada:** Utiliza um *dataset* não anotado, e busca de alguma forma colocar ordem nos dados, que pode ser feito por meio de agrupamento, e assim descobrir as características que definem cada grupo provenientes da entrada de dados.
- **Semi-Supervisionada:** Utiliza um *dataset* mesclado, contendo dados anotados e não anotados, tendo o mesmo objetivo do aprendizado supervisionado, mas busca fazer com que o aprendizado extraído dos dados não anotados atinja um modelo melhor.

3.2 Redes Neurais Artificiais

De uma forma geral, uma RNA é uma implementação que busca modelar o modo como o cérebro executa uma tarefa ou função de interesse [13]. As RNA empregam um

grande número de interconexões de células de computação simples, referidas como neurônios ou unidades de processamento, como pode ser visto na Figura 3.1. Segundo o autor Simon Haykin[13], podemos definir Redes Neurais Artificiais como:

> "RNA é um grande processador paralelamente distribuído, feito de simples unidades de processamento, que está naturalmente propensa à guardar conhecimento experimental, e torná-lo disponível para uso."

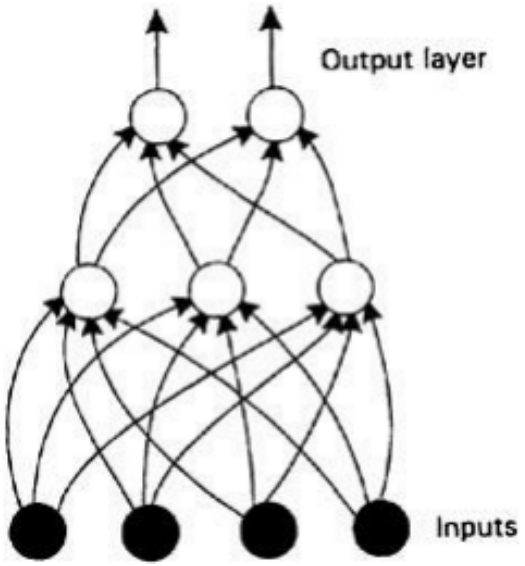

Figura 3.1 – Exemplo de uma Rede Neural Artificial [13]

Os principais benefícios de RNA são, a estrutura paralela distribuída, a capacidade de aprender e com isso generalizar. A generalização permite que RNA produzam saídas, para entradas que não foram encontradas durante o treino. Ainda existem diferentes classes de RNA, mas para os fins deste trabalho, apenas Redes Neurais Recorrentes (RNR) e Redes Neurais Convolucionais (RNC) serão descritas.

3.3 Redes Neurais Recorrentes

Segundo o autor David Kriesel [18], RNR são redes capazes se auto influenciar, utilizando recorrência, isto é, utilizando os dados de computações anteriores para as próximas computações como pode ser visto na Figura 3.2. Este é um dos principais motivos para a utilização deste tipo de RNA em PLN, já que ao utilizar palavras de uma sentença como entrada, cada uma das computações tem influência sobre a próxima.

Ainda segundo o autor David Kriesel [18], um dos problemas encontrados em RNR, é o encolhimento do gradiente, que é o valor utilizado para atualizar os pesos da

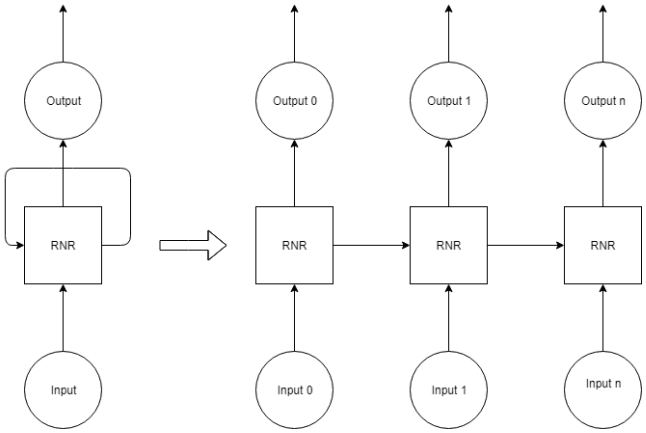

Figura 3.2 – Exemplo de RNR e visualização da abertura de recorrência

RNA. Onde durante o processo de *backpropagation*, os valores vão diminuindo à ponto de tornarem-se insignificantes. Este problema também é chamado de *Short-Term Memory*(STM) já que ele pode acontecer quando a entrada é composta por uma sentença muito longa.

3.3.1 Long Short-Term Memory

Em casos que podem ocorrer o problema de STM, é comum utilizar um tipo especial de RNR, que são as redes *Long Short-Term Memory* (LSTM), inicialmente introduzidas por Sepp Hochreiter e Jurgen Schmidhuber [16]. A LSTM é composta basicamente por uma célula de memória e algumas camadas adicionais de processamento, que podem variar de acordo com a tarefa a ser resolvida, mas o padrão é camada de esquecimento, camada de entrada e camada de saída [10]. Onde após a passagem de cada camada, a célula de memória é atualizada, seja removendo, atualizando ou adicionando dados, para que não ocorra a estagnação que ocorre em uma RNR padrão.

3.4 Redes Neurais Convolucionais

Outra classe de RNA que deve ser mencionada são as RNC. Tal classe de RNA pode ser descrita como especializada em processamento de dados que tem uma topologia de *grid* [11], sendo bastante utilizada em análise e processamento de imagens digitais, mas recentemente já foi utilizada para tarefas de linguagem natural.

A RNC tem como principais características interações esparsas, compartilhamento de parâmetros e representações equivariantes. Além de conforme indicado pelo seu nome,

esta rede emprega em pelo menos uma de suas camadas a operação matemática de convolução.

3.5 Processamento de Linguagem Natural

Linguagem natural é a forma primária com que os humanos se comunicam entre si [7], e PLN refere-se ao uso de métodos computacionais para analisar e processar esta linguagem escrita ou falada. Este processamento como pode ser visto na Figura 3.3, normalmente consiste de uma *pipeline* de ações sobre os dados de entrada, onde cada uma destas etapas tem um propósito definido.

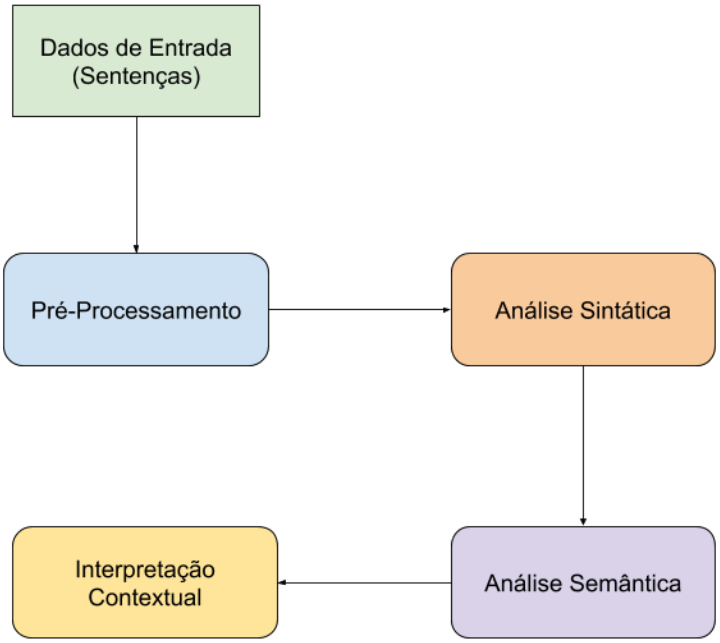

Figura 3.3 – Exemplo de *pipeline* de PLN

O padrão de pipeline de PLN é composto pelas seguintes etapas:

- **Pré-Processamento:** Tratamento inicial dos dados de entrada, como *tokenization*, normalização, extração de informações, entre outros.
- **Análise Sintática:** Determinação da estrutura do texto, tendo como base regras sintáticas.
- **Análise Semântica:** Determinação do significado de palavras, frases e textos.
- **Interpretação Contextual:** A interpretação de qual contexto os dados de entrada se adéquam melhor.

3.6 Extração de Informação

EI é uma tarefa de alto nível de PLN [23], que utiliza métodos computacionais para extrair informações relevantes de textos desestruturados, e transforma-las em dados significativos para contextos computacionais. Esta extração sempre segue critérios, que indicam quais características são importantes de serem extraídas, e qual a forma que ela será representada.

Dois dos principais alvos de EI, devido ao valor das informações contidas neles são:

- **Entidades:** pessoas, locais, lugares.
- **Relacionamentos:** entre entidades.

Assim como em PLN,podemos considerar a arquitetura de EI uma *pipeline*, como pode ser visto um exemplo na Figura 3.4, onde a cada ação podemos refinar mais a informação que está sendo extraída. A seguir temos os passos que normalmente esta *pipeline* segue:

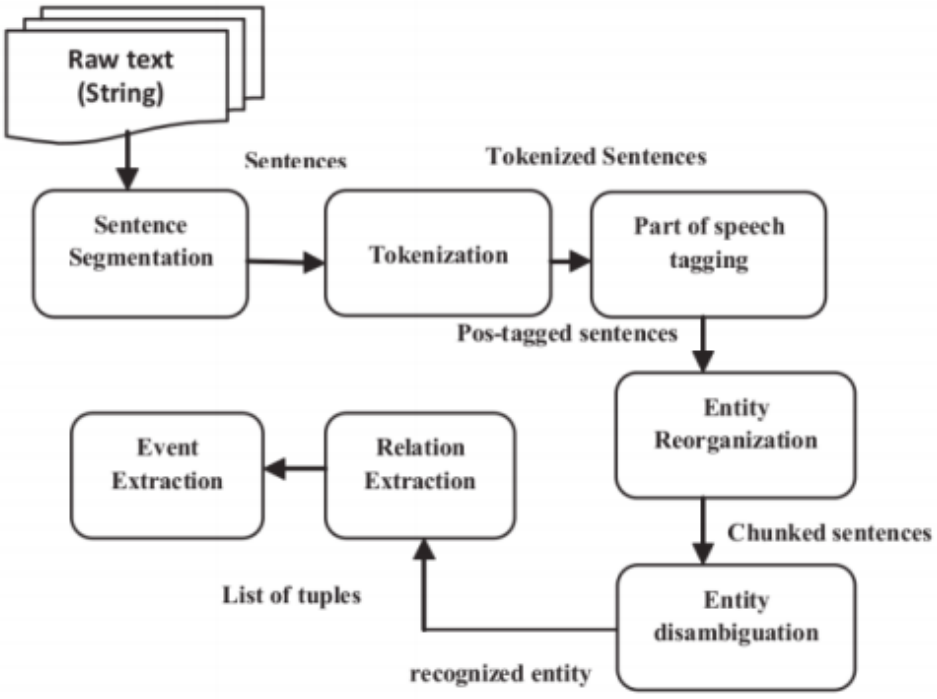

Figura 3.4 – Exemplo de *pipeline* de EI [23]

O padrão de pipeline de PLN é composto pelas seguintes etapas:

- **Parts-of-Speech(POS) Tagging**
- **Parseamento**

- **Reconhecimento de Entidades Nomeadas**
- **Vinculação de Entidades Nomeadas**
- **Resolução de Correferência**
- **Extração de Informação Temporal**
- **Extração de Relacionamento**

Porém é necessário enfatizar que nem sempre é preciso passar por todos estes passos, o que irá definir as ações a serem tomadas é o tipo de informação que está sendo extraída. Dentre as tarefas listadas, REN e ER serão mais profundamente utilizadas neste trabalho.

3.7 Reconhecimento de Entidades Nomeadas

REN é a tarefa de encontrar entidades nomeadas em uma entrada de texto, as seguintes entidades são consideradas nomeadas, onde temos na Tabela 3.1 alguns exemplos de entidades nomeadas em algumas frases:

- **Pessoas**
- **Organizações**
- **Localidades**
- **Organizações Geo-Politicas**
- **Tempo**
- **Unidades Monetárias**

Tabela 3.1 – Exemplos de entidades nomeadas e suas categorizações

Tipo	Tag	Amostra das categorias	Frases
Pessoa	PER	Pessoas, Personagens	**Turing** é o pai da computação
Organização	ORG	Organizações, Times	O **Grêmio** é o rei de copas
Local	LOC	Regiões, Ruas	**Porto Alegre** é demais
Veículo	VEH	Veículos locomotores	O **Fiat Uno** é um carro muito versátil

Além de encontrar estas entidades, é necessário classificá-las no grupo ao qual pertencem, o que acaba tornando a tarefa mais complexa. Visto que nas entidades encontradas pode haver ambiguidade, já que uma pessoa pode dar nome a um local, como

aeroportos, ruas ou até mesmo objetos, como podemos ver na Tabela 3.2. Outro ponto é a definição de quais entradas são entidades e quais não são, e qual é o limite de uma entidade.

Tabela 3.2 – Exemplos de ambiguidade na extração de entidades nomeadas

Entidade	Possíveis Categorias
Assis Brasil	Pessoa, Local
Liverpool	Local, Organização
Oderich	Pessoa, Organização

A Rotulação Sequencial de Palavras é o algoritmo padrão para lidar com a tarefa de REN, onde todas a palavras são rotuladas tanto quanto ao seu tipo e limite. Dentre as famílias de algoritmos que conseguem realizar esta tarefa de rotulação, os mais utilizados são:

- **Algoritmos de Seleção de Características**
- **Redes Neurais**

3.7.1 Algoritmos de Seleção de Características

Nesta abordagem é realizada a extração de características dos dados de entrada, seguido do treinamento de um *Maximum-Entropy Markov Model* (MEMM) ou um modelo sequencial *Conditional Random Fields* (CRF) [17]. Algumas das características procuradas, são:

- **Prefixos**
- **Sufixos**
- **Palavras totalmente em letras maiúsculas**
- **Presença de hífen**
- **Presença de índice de topônimos**

3.7.2 Redes Neurais

O algoritmo neural padrão utilizado nesta abordagem é o LSTM. Para a realização do REN, é necessário utilizar BiLSTM, onde a entrada é passada por um LSTM da esquerda

para a direita, além de um LSTM da direita para a esquerda, onde ao final é realizada uma concatenação dos resultados, e é passado para mais uma camada de CRF, para só assim resultar na rotulação dos dados [17].

3.8 Extração de Relacionamento

ER é a tarefa de encontrar e classificar ou extrair o relacionamento de entidade nomeadas presentes na entrada de dados fornecida [17]. Temos uma infinidade de relacionamentos possíveis entre as entidades, mas os mais comuns são os seguintes:

- **Parentesco:** entre pessoas
- **Empregatício:** entre pessoa e organização
- **Nascimento:** entre pessoa e localidade
- **Propriedade:** entre pessoa e objeto
- **Localização:** entre organização e localidade

Podemos utilizar o modelo teórico de conjuntos para definir estes relacionamentos, onde relacionamento pode ser definido como um conjunto de tuplas de elementos ordenados sobre um domínio [17], os elementos são as entidades nomeadas, e o domínio é o texto previamente tratado em tarefas anteriores da EI, como podemos ver o exemplo na Tabela 3.3.

Tabela 3.3 – Exemplo de Relacionamento de Entidades Nomeadas

Relacionamento	Frase
programador de	**José** programador da **CWI** está desenvolvendo um *software*
filho de	**José** é o filho mais novo de **João**
fundadora de	**Maria** é a fundadora da *fintech* multinacional **Finmoney**

Assim como na tarefa de REN, temos diferentes técnicas para realizar ER, e a seguir temos um compilado de algumas delas:

- **Uso de Padrões**
- **Aprendizado Supervisionado**
- **Aprendizado Semi-Supervisionado**
- **Aprendizado Não Supervisionado**

3.8.1 Uso de Padrões

Utiliza-se de algoritmos de reconhecimento de padrões léxico-sintáticos, primeiramente desenvolvido por Marti A. Hearst [14], é um dos primeiros algoritmos usados para REN. Os padrões tem de ser criados manualmente, o que por um lado trás benefícios ao aplicar em domínios fechados de entrada, mas torna-se extremamente custoso ao tentar torná-lo mais genérico.

3.8.2 Aprendizado Supervisionado

Ao utilizar o Aprendizado de Máquina Supervisionado, é necessário passar por uma série de etapas. Primeiramente deve-se definir um conjunto de entidades e relacionamentos, após isso um *corpus* é anotado manualmente para ser utilizado como entrada de treinamento, e finalmente utilizamos uma parte separada da entrada de dados anotada como teste. Após a etapa de treinamento, é feito o teste, onde dependendo do objetivo da tarefa, os relacionamentos são classificados dentro de um conjunto estabelecido de possíveis relacionamentos, ou as palavras que formam um relacionamento entre as entidades são extraídas.

Se o conjunto de treino for suficientemente grande, ou todas as entradas se encaixem no mesmo gênero, este método tem uma boa precisão. Mas devido ao grande custo de rotular um grande *dataset* de treino, a tarefa de REN é mais propensa a Aprendizado Semi-Supervisionado ou Aprendizado Não-Supervisionado.

3.8.3 Aprendizado Semi-Supervisionado

Em casos que a quantidade de dados de treino não é muito extensa, mas nós temos alguns padrões iniciais confiáveis, podemos realizar um processo de agregação, conhecido na área da computação como *bootstrapping*, e gerar novos dados para utilizar no classificador [17]. Onde a partir dos padrões iniciais, eles são procurados nos dados de entrada, e onde são encontrados, extraímos e generalizamos o contexto, desta forma é possível aprender novos padrões para procurar.

Este método ajudar a evitar o *overfitting*, podendo levar a classificação correta de dados mais genéricos. Porém a entrada de padrões iniciais que tenham uma confiabilidade baixa, pode levar à uma precisão ruim, visto que os novos padrões aprendidos não serão muito bons para a tarefa.

3.8.4 Aprendizado Não-Supervisionado

Esta abordagem tem como principal objetivo, realizar a tarefa de ER usando textos puros, sem nenhuma anotação como as utilizadas nas seções anteriores, e ainda sem lista prévia de relacionamentos [17]. Esta tarefa é chamada de Extração de Relacionamentos Abertos, e é normalmente utilizada para textos retirados de páginas web [8].

Sua maior vantagem é a capacidade de lidar com um número extremamente grande de relacionamentos, mas como contraponto necessita de uma estrutura maior para realizar este mapeamento dinâmico. Atualmente a maioria dos métodos utilizados para a Extração de Relacionamentos Abertos, baseia-se fortemente em verbos, o que pode levar à uma precariedade na extração de relacionamentos que não são expressados através de verbos.

4. TRABALHOS RELACIONADOS

Neste capitulo serão detalhados alguns trabalhos relacionados à proposta deste TCC. Apesar de alguns materiais aqui apresentados não serem voltados para a língua portuguesa, ajudam a compreender as abordagens que estão sendo utilizadas neste momento para realizar as tarefas de PLN, mais especificamente na área de ER.

Dentre os trabalhos relacionados selecionados, buscou-se a escolha de diferentes abordagens de RNA, um trabalho que foi submetido ao fórum *IberLEF* 2019, e um sistema já existente capaz de realizar a tarefa de ER. De forma que seja possível verificar as principais características, aspectos positivos, além dos resultados obtidos por cada um deles, e desta forma viabilizar que durante este trabalho seja viável desenvolver um sistema que possa alcançar resultados satisfatórios para a tarefa proposta.

4.1 Extração de Relacionamentos utilizando Redes Neurais Convolucionais

Sistema desenvolvido pelo *Massachusetts Institute of Technology* (MIT) para *International Workshop on Semantic Evaluation 2017* (SemEval-2017) [19]. Este sistema foi implementado para resolver a tarefa 10, proposta no *workshop*, que consistia da extração de palavras chaves e relacionamentos, dentro de publicações científicas. Onde inclusive a solução proposta ficou em primeiro lugar.

O modelo desenvolvido é composto por três partes, sendo elas:

- **Pré-processamento**
- **RNC**
- **Pós-processamento**

Na etapa de pré-processamento, é recebida uma entrada de texto, contento um parágrafo de uma publicação, junto da lista de entidades e a sua localização no texto. Como saída, temos os seguintes itens, a posição relativa das entidades relacionadas, seus tipos e o POS *Tag*. Ainda são feitos alguns tratamentos, de modo que informações desnecessárias são removidas, tornando os dados mais limpos.

A RNC por sua vez recebe a saída de dados que foi processada, e realiza a predição do tipo de relacionamento entre as entidades presentes na entrada. Existem 4 camadas na RNC, que são elas, camada de *embedding*, que converte cada uma das características extraídas na etapa anterior em um vetor de *embedding*, a camada convolucional utiliza a unidade linear retificada como função de ativação, para transformar os vetores de *embedding* em características mapeadas, a camada de *Max-pooling* seleciona a característica

com maior valor do mapa, e por fim a camada totalmente conectada utilizando a a função *softmax* de ativação retorna a probabilidade de cada relacionamento.

E a ultima etapa, o pós-processamento, utiliza regras para corrigir a saída de dados da RNC, onde estas regras são baseadas nos exemplos fornecidos para o treinamento, para que sejam o mais fiel com o senso comum.

O *dataset* utilizado foi o *ScienceIE* [5], que consiste em 500 artigos científicos. Sendo estes artigos divididos igualmente entre os seguintes domínios, ciência da computação, ciências materiais e física. Os seguintes relacionamentos eram procurados nos artigos, sinônimo de, e hipônimo de.

A Tabela 4.1 mostra os resultados obtidos, onde é possível notar uma grande diferença na extração dos dois tipos de relacionamentos diferentes.

Tabela 4.1 – Resultados obtidos utilizando RNC

Relação	Precisão	Recall	F-Measure
Sinônimo de	0.820	0.813	0.816
Hipônimo de	0.455	0.421	0.421

4.2 Classificação de Relacionamentos utilizando Redes Neurais Recorrentes

Este trabalho busca demonstrar que sistemas baseados em modelos RNR podem ser capazes de obter uma performance melhor em classificação de relacionamentos, do que sistemas baseados em modelos RNC [25]. Foi utilizado o *SemEval-2010 Task 8 dataset* [15], onde são observados 9 tipos de relacionamentos, e tem como base textos extraídos da *web*, onde estão anotadas sempre duas entidades nomeadas por sentença.

Esta RNR é composta de 3 camadas, e assim como o modelo apresentado cada uma tem uma função específica. Sendo elas, a primeira camada realiza a transformação de *word embedding*, que mapeia cada uma das palavras de uma sentença em um *embedding* de baixa dimensionalidade. Uma camada bidirecional recorrente vem em seguida, sendo utilizada para fazer predições tanto de palavras anteriores como posteriores a palavra analisada no momento, resultando em características a nível de palavra, também chamada de representação. A ultima camada é o *Max-Pooling*, que mescla as características a nível de palavra, de cada palavra analisada na sentença, baseado no valor máximo de cada característica de cada dimensão extraída na primeira camada, em um vetor com características a nível de sentença, onde então é classificada a relação entre entidades utilizando este vetor final.

Para fins de comparação, foi utilizado o trabalho de Zeng et al [24], então na Tabela 4.2, podemos ver um comparativo de um sistema baseado em RNC, e o sistema baseado em RNR. É possível ver que apesar de não muito significativo, o RNR conseguiu um desempenho melhor que o modelo baseado em RNC.

Tabela 4.2 – Comparativo entre RNC e RNR utilizando SemEval-2010 Task 8 dataset

Modelo	F-Measure
RNC	0.789
RNR	0.796

4.3 NLPyPort: Reconhecimento de Entidades Nomeadas com CRF e Extração de Relacionamento Baseado em Regras

O trabalho de Hugo Gonçalo Oliveira et al [9] é um sistema capaz de realizar algumas tarefas da área de PLN, com REN e ER. Este sistema participou do fórum *IberLEF* 2019, realizando as tarefas de REN e ER, propostas no fórum. Para a tarefa de REN, é utilizado um modelo CRF, enquanto para a tarefa de ER, é utilizado um método baseado em regras.

Apesar de utilizar uma abordagem bem diferente da proposta para o sistema desenvolvido neste trabalho, ele será utilizado como referência para comparação de resultados. Esta escolha é motivada pelo fato deste sistema ser o único participante da tarefa de ER do fórum *IberLEF* 2019.

As regras do sistema são ordenadas conforme a especificidade e abrangência do relacionamento buscado, como pode ser visto na Figura 4.1, que trás exemplos de uma regra muito específica, e uma regra muito abrangente. As regras são divididas em três partes, e cada uma delas é dividida pelo carácter especial *&*. A primeira parte é o ranqueamento da regra, a segunda é uma representação binária de cada palavra presente na sentença, onde o valor 1 representa palavras que são parte do relacionamento entre o par de entidades, e o valor 0 representa as palavras que não fazem parte do relacionamento, e a ultima parte da regra é formada pelo POS *Tag* de cada palavra, já que esta informação traria melhores resultados que a utilização da própria palavra.

```
                Regra com alta especificidade
1&0101000001111&v prp  v prp art n n punc punc prp art n

                  Regra muito abrangente
                         1&1&punc
```

Figura 4.1 – Exemplo de regras do sistema NLPyPort

A Tabela 4.3 mostra os resultados obtidos utilizando apenas as relações extraídas que estão completamente corretas, enquanto a Tabela 4.4 mostra os resultados do sistema quando as relações que estão parcialmente corretas são adicionadas ao calculo. Apesar dos bons resultados obtidos, os autores fizeram ressalvas quanto aos relacionamentos extraídos, e qual seria a real utilização deles.

Tabela 4.3 – Resultado da extração de relacionamentos completamente corretos

Precisão	Recall	F-Measure
0.736	0.711	0.7235

Tabela 4.4 – Resultado da extração de relacionamentos utilizando relacionamentos parcialmente corretos

Precisão	Recall	F-Measure
0.7662	0.748	0.757

4.4 RelP: Portuguese Open Relation Extraction

O RelP é um sistema capaz de realizar a extração de relacionamentos abertos em textos de língua portuguesa [8]. Este sistema realiza a extração de descritores de relacionamentos explícitos entre duas ENs, utilizando CRF. Devido aos resultados de estado da arte alcançados pelo RelP na extração de relacionamentos abertos, onde utilizando um subconjunto da *Golden Collections* das duas conferências HAREM obteve *F-measure* em torno de 60% em relacionamentos entre pessoas, locais e organizações, ele também será utilizado como referência para comparação dos resultados obtidos no desenvolvimento deste trabalho.

Este sistema primeiramente realiza o pré-processamento das entradas, onde é feito o POS *tagging* das palavras presentes na sentença, anotação sintática e semântica,

a verificação da presença das palavras em um dicionário externo, categorização de EN e necessidade de geração de *feature*. Todos estes dados coletados na etapa de pré-processamento, são transformados em um vetor de *features* que pode ser observado na Figura 4.2, que por fim é utilizado para gerar o modelo CRF, atribuindo certos pesos para as *features* extraídas, e assim ser capaz de indicar palavras que compõem um relacionamento entre ENs. Como saída do sistema, é feita a tripla de informações, contendo as duas EN junto das palavras que descrevem o relacionamento entre elas, como mostra a Figura 4.3.

Word	Canonic form	Semantic	POS	Syntactic	NE
[…]					
Ronaldo_Lemos	Ronaldo_Lemos	<hum>	PROP	@SUBJ>	PERS
,					
diretor	diretor	<Hprof>	N	@N<PRED	
de	de		PRP	@N<	
a	o		DET	@>N	
Creative_Commons	Creative_Commons	<org>	PROP	@P<	ORG
[…]					

Figura 4.2 – Exemplo de Vetor de *Features* [8]

(Ronaldo_Lemos<O>, diretor<I-REL> de<I-REL>,
Creative_Commons<O>)

Figura 4.3 – Exemplo de Tripla de Saída do RelP

5. RECURSOS NECESSÁRIOS

5.1 Python

Python é uma linguagem de programação de alto nível, interpretada, interativa e orientada a objetos [2]. Sua principal característica é o poder de processamento, junto a uma sintaxe muito clara e de fácil compreensão.

Ela é vista como uma linguagem de programação de propósito geral, sendo assim ela pode ser utilizada para diferentes classes de problemas. Além disso ela vem com uma vasta biblioteca padrão, que pode ser utilizada em inúmeras áreas, tais como processamento de textos, protocolos de internet, interface para sistemas operacionais, entre outras.

A motivação para a escolha desta linguagem é a curva de aprendizado acessível para o tempo necessário de desenvolvimento da tarefa, e também conta com uma comunidade muita ativa, e atenciosa quanto a auxiliar com questões quanto a linguagem em si, ou as diversas bibliotecas que existem. Inclusive a utilização de *Python* para esta área de IA está ligada com número de bibliotecas disponíveis para tal propósito, abaixo estão listadas algumas delas.

- ***TensorFlow***
- ***Pytorch***
- ***Keras***
- ***scikit-learn***

Além das bibliotecas voltadas para IA de uma forma geral, temos algumas que são direcionadas especificamente para PLN. Apesar da maior parte das funcionalidades, serem mais sofisticadas quando utilizadas em textos de língua inglesa, elas possuem suporte para inúmeras outras línguas, incluindo a portuguesa, que é o foco deste trabalho. Segue a lista das principais:

- ***Natural Language Tool Kit - NLTK***
- ***spaCy***

5.2 Google Colaboratory

Google Colaboratory é uma ferramenta de pesquisa online para realizar experiências na área de Aprendizado de Máquina [1]. Esta ferramenta é baseada em um ambiente

Jupyter notebook, que é uma aplicação *web open-source*, que possibilita a criação e compartilhamento de documentos que contem código.

Apesar do *Jupyter notebook* suportar mais de 40 linguagens de programação, o *Google Colaboratory* atualmente suporta apenas *Python 2.6* e *Python 3.6*. Porém a grande vantagem desta ferramenta é a possibilidade de utilizar uma GPU, o que traz muitos benefícios durante o treinamento e teste de sistemas. A ferramenta ainda conta com as bibliotecas *PyTorch*, *TensorFlow*, *Keras*, que foram citadas anteriormente, entre outras. Tudo isso é disponibilizado de forma gratuita, tornando o *Google Colaboratory* ainda mais atrativo para o desenvolvimento de pesquisa na área de Aprendizado de Máquina.

5.3 NILC Word Embeddings

Aplicações utilizadas para PLN normalmente utilizam palavras como dados de entrada, o que pode não trazer tanta significância direta ao sistema utilizado, para tratar isto, é necessário fornecer representações que tragam forneçam algum significados aos dados que stão sendo utilizados como entrada, e uma destas representações é *word embeddings*, que transforma a palavra em um vetor de baixa dimensionalidade, que é convencionado conter informações semânticas e sintáticas da palavra [20]. Para criar um modelo de *word embeddings* é necessário passar um *corpus* por algum algoritmo de geração de *embeddings* [12], onde os mais comuns são *FastText, GloVe, Wang2Vec* e *Word2Vec*.

Para evitar esta etapa de treinamento de um modelo de *word embeddings*, será utilizado o trabalho feito por Hartmann et al [12],onde foram treinados 31 modelos de *word embeddings*, utilizando os 4 algoritmos mencionados anteriormente. Estes modelos foram treinados utilizando um grande *corpus* contendo textos em português do Brasil e de Portugal, os modelos treinados apresentam variações nas dimensões, que iniciam em 50 dimensões, podendo alcançar 1000 dimensões.

5.4 Recursos de Hardware

5.4.1 Desktop

- Processador: Intel Core i7-5820K 3.30GHz
- Memória RAM: 16GB
- Unidade de Processamento Gráfico: NVIDIA GeForce GTX 980 4GB

5.4.2 Notebook

- Processador: Intel Core i5-3230M 2.6GHz
- Memória RAM: 8GB
- Unidade de Processamento Gráfico: AMD Radeon HD 8850M 2GB

6. IBERLEF 2019

IberLEF (Iberian Languages Evaluation Forum) é o fórum de avaliação de línguas ibéricas, resultante da união de outros dois *workshops* de avaliação, que são o TASS (Taller de Análisis Semántico en la SEPLN) e o *IberEval*. O principal objetivo do *IberLEF* é encorajar a comunidade de pesquisas a organizar tarefas de processamento e entendimento de textos, de uma forma competitiva, com intuito de definir novos desafios em campos de pesquisa e alcançar novos padrões de estado da arte para PLN, que envolva pelo menos uma das seguintes línguas ibéricas: espanhol, português, catalão, basco e galego.

No ano de 2019 o *IberLEF* conta com 9 tarefas, sendo elas análise de humor baseado em anotações humanas, análise de sentimento, detecção de ironia, REN e ER. A maior parte das tarefas está direcionada para a língua espanhola, porém as tarefas de REN e ER são voltadas para a língua portuguesa, tornando esta tarefa ideal para o trabalho desenvolvido neste documento, visto que serão enviados sistemas capazes de realiza-la, e assim será possível realizar a comparação de resultados.

6.1 IberLEF 2019 - Reconhecimento de Entidades Nomeadas e Extração de Relacionamentos em Português

Esta tarefa proposta no *IberLEF* 2019, é compostas de 3 sub-tarefas que são da área de PLN. São elas:

- **Tarefa 1: Reconhecimento de Entidades Nomeadas**
- **Tarefa 2: Extração de Relacionamentos**
- **Tarefa 3: Extração de Relacionamentos Abertos Gerais**

Dentre elas, a tarefa 2 foi a escolhida como foco deste trabalho, visto que ela é a mais promissora para a implementação do sistema utilizando o modelo RNR.

6.1.1 Tarefa 2 - Extração de Relacionamentos

A tarefa de ER apresentada neste fórum propõem a criação de um sistema, que seja capaz de realizar a extração automática de qualquer descritor de relacionamento presente entre um par de entidades nomeadas. Estas entidades se encaixam nas categorias de Pessoa, Lugar ou Organização.

Foram definidas 3 fases distintas para o desenvolvimento da tarefa:

- **Fase de Desenvolvimento do Sistema:** Desenvolvimento do sistema utilizando um pequeno *dataset* anotado pelos coordenadores da tarefa.
- **Fase de Teste do Sistema:** Teste do sistema utilizando um *dataset* de teste fornecido pelos coordenadores da tarefa.
- **Fase de Avaliação do Sistema**: Avaliação do sistema, utilizando as métricas de precisão, *recall* e *f-measure*.

6.1.2 Dataset

É possível observar na Figura 6.1 alguns exemplos de sentenças presentes no *dataset* fornecido pelos coordenadores da tarefa, onde as entidades estão grifadas na sentença original, e existe um tripla indicando o par de entidades e o relacionamento extraído.

Examples
(1) A **Marfinite** fica em o **Brasil**. (**Marfinite** is located in **Brasil**.) Relation: fica em (located in) Triple: (Marfinite, fica em, Brasil)
(2) Os aparelhos regressaram à base na **Turquia**, acrescenta o comunicado do **Pentágono**. (The equipment returned to the base in **Turquia**, added the statement of **Pentágono**.) Relation: no relation
(3) A **Marfinite** abre perspectivas de negócios através de novos distribuidores em o **Brasil**. (A **Marfinite** opens business perspectives through new distributors in **Brasil**). Relation: abre perspectivas em (opens perspectives in) Triple: (Marfinite, abre perspectivas em, Brasil)
(4) **Ronaldo_Goldone** continua atuando em as atividades de o **Niterói_Rugby**. (**Ronaldo_Goldone** continues to work in the activities of **Niterói_Rugby**.) Relation: atuando em (to work in) Triple: (Ronaldo_Goldone, atuando em, Niterói_Rugby)
(5) **Hugo_Doménech**, professor de a **Universidade_Jaume_de_Castellón**. (**Hugo_Doménech**, teacher of **Universidade_Jaume_de_Castellón**.) Relation: professor de (teacher of) Triple: (Hugo_Doménech, professor de, Universidade_Jaume_de_Castellón)
(6) Em 1956, **Amílcar_Cabral** criou o **Partido_Africano**. (In 1956, **Amílcar_Cabral** created the **Partido_Africano**.) Relation: criou (created) Triple: (Amílcar_Cabral, criou, Partido_Africano)
(7) **António_Fontes** de a **AIPAN**. (**António_Fontes** of the **AIPAN**.) Relation: de (of) Triple: (António_Fontes, de, AIPAN)
(8) A **USP** (**Universidade_de_São_Paulo**) aprovou a iniciativa dos alunos. (**USP** (**University_of_São_Paulo**) approved the students' initiative.) Relation: (Triple: (USP, (, Universidade_de_São_Paulo)
(9) O **Presidente** em exercício de o **Conselho**. (The current **Presidente** of the **Conselho**.) Relation: de (of) Triple: (Presidente, de, Conselho)
(10) A **Legião_da_Boa_Vontade** comemora o aniversário da sua implantação em **Portugal**. (**Legião_da_Boa_Vontade** celebrates the birthday of its establishment in **Portugal**.) Relation: implantação em (establishment in) Triple: (Legião_da_Boa_Vontade, implantação em, Portugal)

Figura 6.1 – Exemplo de *dataset* de treino do IberLEF 2019

A seguir serão apresentadas algumas tabelas contendo informações sobre os *datasets*, tanto de teste quanto de treino, que podem ser relevantes para compreender os

dados que estão sendo utilizados, e quais são as suas principais características. Além de auxiliar no entendimento dos resultados obtidos pelo sistema.

Na Tabela 6.1 podemos ver o tamanho de cada um dos *datasets*, ou seja quantas sentenças cada um deles possui. Desta observação, podemos concluir que é um número bem limitado de dados para realizar o treinamento do sistema, e que as características que serão utilizadas como entrada podem ter um papel fundamental para um bom desempenho.

Tabela 6.1 – Tamanho dos *datasets*

Dataset	Número de Sentenças
Treino	90
Teste	149

Com a Tabela 6.2 podemos visualizar as categorias de entidades, e como elas estão identificadas no *dataset*. Já a Tabela 6.3 nos mostra a distribuição de categorias de entidades em cada um dos *datasets*, enquanto a Tabela 6.4 mostra o número de relacionamentos entre as categorias de entidades. Destes dados podemos perceber que o número de organizações é bem superior as outras duas categorias, o que pode trazer dificuldades de aprendizado com estas categorias.

Tabela 6.2 – Categorias de entidades nos *datasets*

Categoria da Entidade	Representação no *dataset*
Pessoa	PER
Lugar	PLC
Organização	ORG

Tabela 6.3 – Distribuição de entidades nos *datasets*

Categoria da Entidade	*Dataset* Treino	*Dataset* Teste
PER	31	56
PLC	30	46
ORG	119	196
Total	180	298

A Tabela 6.5 mostra as 5 relações completas (conjunto de palavras dentro de uma relação) que mais aparecem no *dataset* de teste, e na Tabela 6.6 temos as 5 relações completas que mais aparecem no *dataset* de treino. Diferente de muitas aplicações de ER presentes na literatura sobre PLN, onde existe um conjunto fechado de relacionamentos em que as entidades podem se enquadrar, como por exemplo, filiado a, pertence a, esta tarefa

Tabela 6.4 – Número de relacionamentos entre entidades

Tupla de Entidades	*Dataset* Treino	*Dataset* Teste
ORG-ORG	29	47
ORG-PLC	25	37
PER-ORG	24	53
ORG-PER	7	3
PLC-ORG	5	9
Total	90	149

está aberta quanto aos relacionamentos, isto é, são apenas um conjunto de palavras que caracterizam o relacionamento entre entidades, o que torna a tarefa mais complexa.

Tabela 6.5 – 5 Relações completas que mais aparecem no *dataset* de treino

Relação	Número de vezes em que aparece no *dataset*
de	14
em	13
subvive de	5
(	3
mandar em	2

Tabela 6.6 – 5 Relações completas que mais aparecem no *dataset* de teste

Relação	Número de vezes em que aparece no *dataset*
de	43
(	18
em	13
pesquisadores de	5
criticado	2

7. PRÉ-PROCESSAMENTO DOS DADOS

Como em todo pipeline de tarefas de PLN, neste trabalho será necessário realizar o pré-processamento dos dados. Este processamento será utilizado para criar as representações de entrada de dados no sistema, assim como a saída de dados que ele fornecerá.

Desta forma teremos as seguintes tarefas, que deverão ser realizadas para obtermos os dados para o sistema:

- Padronizar o tamanho das sentenças
- Transformar palavras das sentenças em identificadores
- Identificar a posição das entidades na sentença
- Realizar o POS Tag em cada palavra da sentença
- Identificar as palavras que são consideradas relacionamentos

7.1 Padronização de Tamanho

É necessário definir um tamanho padrão fixo para as entradas de dados no sistema, para isso foi verificado qual a sentença mais longa presente no *dataset*, e o tamanho dela será o padrão. Após esta verificação todas as sentenças são preenchidas com um *token* especial, <PAD>, que representa a inclusão de dados para preencher a sentença até o tamanho padrão definido.

7.2 Transformação de Palavras em Identificadores

Conhecido como Palavra para Identificador, este é o processo de mapear cada uma das palavras presentes no *dataset* para um identificador. Isto é necessário para simplificar a entrada de dados no sistema, já que os sistemas computacionais tem uma melhor performance ao trabalhar com valores numéricos, ao invés de utilizar as palavras.

Para realizar este processo, é feita uma iteração por todas as palavras de todas as sentenças do *dataset*, onde cada palavra que ainda não tenha sido mapeada recebe um identificador, é importante ressaltar que o identificador 0 fica com o *token* definido anteriormente para padronizar o tamanho das sentenças. Na Figura 7.1 podemos ver um exemplo de como fica uma sentença após o processamento.

```
Sentença original
[O cachorro é o melhor amigo do homem <PAD> <PAD>]

Sentença processada
[1 2 3 1 4 5 6 7 0 0]
```

Figura 7.1 – Exemplo processamento de palavras de uma sentença para identificadores

7.3 Marcação de Posição de Entidades

Em alguns trabalhos relacionados a ER, os sistemas recebem como uma das entradas, um vetor com distâncias relativas de cada palavra para a entidade nomeada, como pode ser visto na Figura 7.2. Portanto no caso da tarefa proposta neste trabalho, seriam necessário dois vetores, já que cada uma das sentenças tem um par de entidades nomeadas.

```
Exemplo de Sentença
[O gaúcho Carlos paleontólogo de PUCRS é reconhecido mundialmente]

Posição Relativa da Entidade Nomeada Carlos
[-2 -1 0 1 2 3 4 5 6]

Posição Relativa da Entidade Nomeada PUCRS
[-5 -4 -3 -2 -1 0 1 2 3]
```

Figura 7.2 – Exemplo de Posição Relativa

Para diminuir o número de entradas no sistema, e simplificar a representação da entrada, será utilizado apenas um vetor binário para indicar a posição das entidades na sentença, onde o valor 1 representa uma entidade nomeada e o valor 0 representa as outras palavras, como ilustra a Figura 7.3.

```
Exemplo de Sentença
[O gaúcho Carlos paleontólogo de PUCRS é reconhecido mundialmente]

Posição das Entidades na Sentença
[0 0 1 0 0 1 0 0 0]
```

Figura 7.3 – Exemplo de Vetor Binário para marcação de posição de Entidades Nomeadas

7.4 POS Tag de Palavras

Visto o tamanho reduzido do *dataset*, é necessário a adição de algumas características adicionais como entrada do sistema, neste caso foi escolhida a definição de POS Tag de cada uma da palavras presentes na sentença. Esta escolha se deve ao fato do sistema ter como objetivo a extração de qualquer palavra que caracterize um relacionamento entre duas entidades, e o POS *Tag* pode ajudar o sistema a aprender melhor como identificar cada uma destas palavras.

Como o sistema não tem enfoque na parte de POS Tag de PLN, será utilizada a biblioteca *spaCy* para realizar marcação de cada palavra, a Tabela 7.1 define os POS *Tags* que a biblioteca *spaCy* identifica. Assim como na etapa de transformação de palavras para identificadores, cada POS Tag irá corresponder a um identificador, criando assim um vetor de valores numéricos, como é possível observar na Figura 7.4.

Tabela 7.1 – Lista de POS Tags disponíveis na biblioteca *spaCy*

POS	Descrição
ADJ	Adjetivo
ADP	Preposição
ADV	Adverbio
AUX	Auxiliar
CCONJ	Conjunção
DET	Artigo
NOUN	Substantivo
NUM	Número
PAD	Token especial
PRON	Pronome
PROPN	Nome Próprio
PUNCT	Pontuação
VERB	Verbo

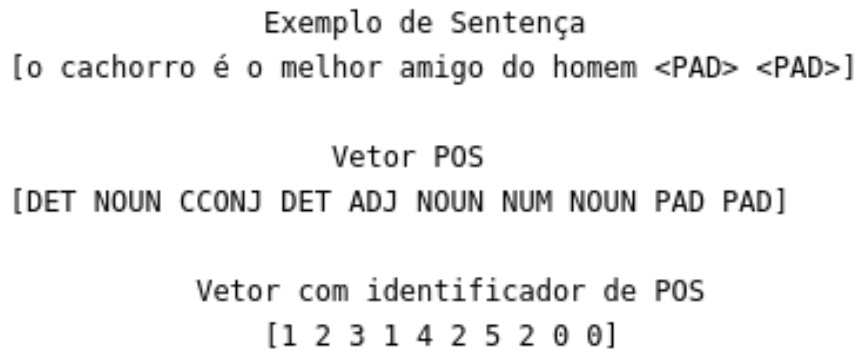

Figura 7.4 – Exemplo de processamento de POS Tag

7.5 Marcação de Relacionamento na Sentença

A saída do sistema serão todas as palavras de um sentença que representam um relacionamento entre duas entidades nomeadas, a forma mais trivial de gerar esta representação é novamente através de um vetor binário. Para isso, as palavras anotadas em cada sentença do *dataset*, que fazem parte do relacionamento serão mapeadas para o valor 1, enquanto as outras serão mapeadas para o valor 0, como exemplificado na Figura 7.5.

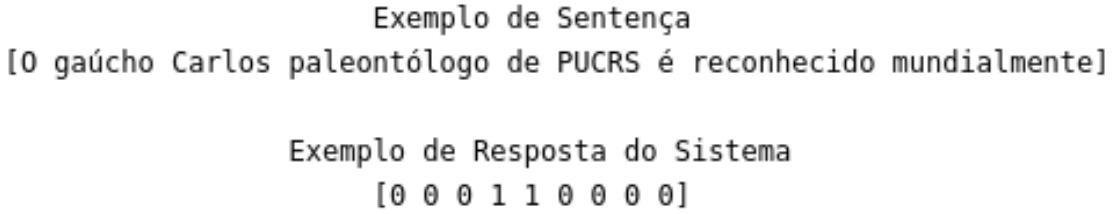

Figura 7.5 – Exemplo de resposta do sistema

Estes dados serão utilizados na parte de verificação de resultados do sistema, tanto treino quanto teste. Já que para verificar o desempenho do sistema, é necessário comparar os resultados fornecidos pelos coordenadores da tarefa, que são os resultados corretos, com as respostas fornecidas pelo sistema.

8. MODELOS DESENVOLVIDOS

Neste capítulo do documento serão apresentados os modelos desenvolvidos, assim como as etapas necessárias para isto. As escolhas de arquitetura também serão esclarecidas quando for necessário.

Foram desenvolvidos modelos utilizando duas arquiteturas diferentes de RNR, as principais arquiteturas disponíveis podem ser observadas na Figura 8.1. Inicialmente a tarefa foi compreendida de forma equivocada, assim sendo escolhida a arquitetura de **Vários para Um**, o que resultou em um modelo classificador de relacionamentos, que após testes rápidos foi descartado devido aos resultados totalmente irrelevantes ao trabalho. Então, após o entendimento completo da tarefa, optou-se pelo modelo **Vários para Vários com Tamanhos Alinhados**, que conseguiria suprir a necessidade de atribuir valores individuais para cada uma das palavras de entrada do modelo.

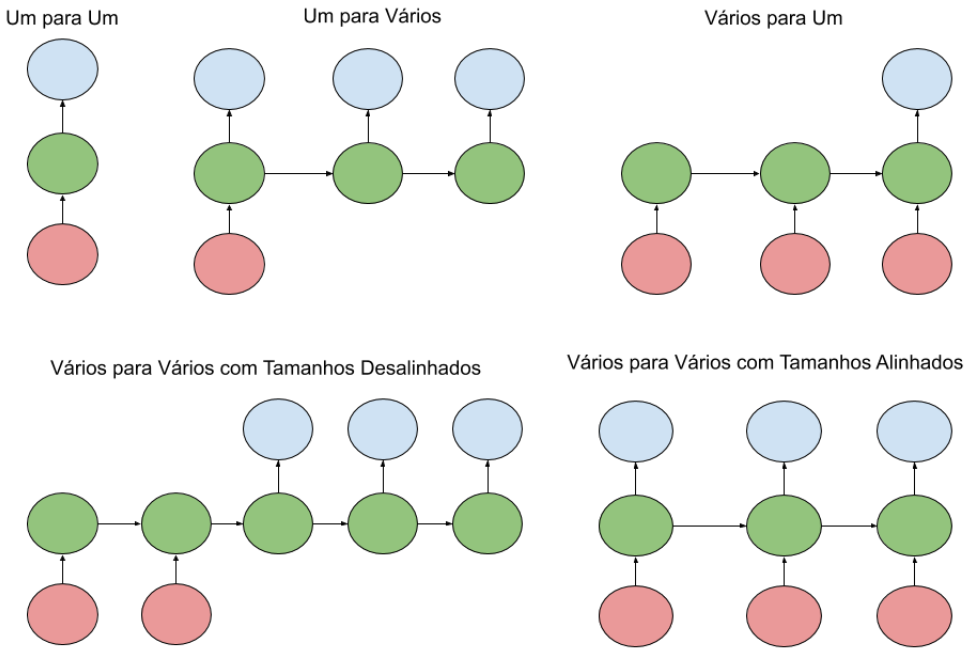

Figura 8.1 – Arquiteturas de RNR

8.1 Modelo Extrator de Relacionamentos

Com a arquitetura correta definida, foi criado um modelo abstrato genérico, como pode ser visto na Figura 8.2. Este modelo genérico serve como base para a criação de diferentes versões de modelos, conforme as entradas que estão sendo usadas, a Tabela 8.1

informa o nome dos modelos que serão criados nesta arquitetura, assim como as entradas. Os nomes apresentados passarão a ser utilizados no restante do trabalho como referência dos modelos.

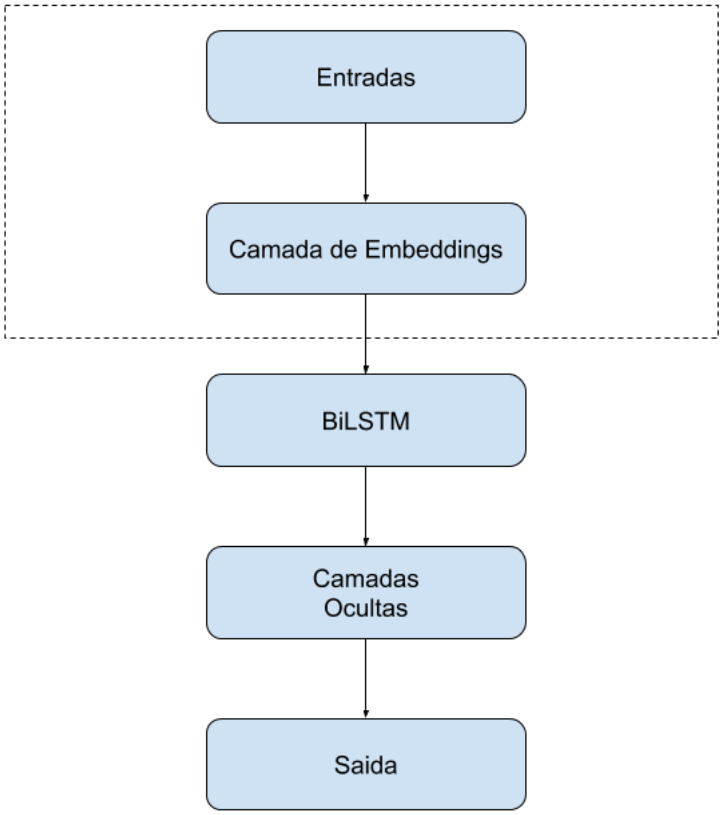

Figura 8.2 – Modelo Extrator de Relacionamentos Genérico

Tabela 8.1 – Descrição de Entrada dos Modelos

Nome do Modelo	Entradas Utilizadas
Modelo Simples	Identificador de Palavras
Modelo Intemediário	Identificador de Palavras + Indicador de EN
Modelo Completo	Identificador de Palavras + Indicador de EN + Tipo de POS Tag

A seguir temos a definição de cada uma das camadas, além do propósito delas neste modelo genérico.

1. **Entradas:** Entradas do modelo, podendo ser uma simples identificação da palavra, como todos os dados coletados na etapa de pré-processamento.
2. **Camada de Embeddings:** Uma junção de *embeddings* de todas as entradas do modelo.

3. **BiLSTM:** Realiza a recorrência da rede tanto da esquerda para a direita, quanto ao inverso.

4. **Camadas Ocultas:** Algumas camadas ocultas do modelo, sendo uma delas de *Dropout*, utilizada para de forma arbitraria ignorar alguns dos neurônios do modelo e assim evitar o *overfitting*.

5. **Saída:** Saída do modelo utilizando a função de ativação Sigmóide, para mapear cada um dos dados de entradas nos valores 0 ou 1, sendo 1 representando palavras que compõem o relacionamento, e 0 palavras que podem ser descartadas.

As duas primeiras camadas do modelo genérico, entradas e *embeddings*, são substituídas nos modelos que foram de fato implementados. Na Figura 8.3 é possível observar a forma sequencial que os dados passam por estas camadas iniciais no **Modelo Simples**. Os **Modelos Intermediário** e **Completo**, Figura 8.4 e Figura 8.5 respectivamente, introduzem a entrada paralela de dados, onde após a passagem pelos *embeddings*, são concatenados em uma representação única, ou seja, o vetor que antes continha apenas as dimensões do *word embeddings*, agora tem anexado ao seu final as dimensões dos outros *embeddings*.

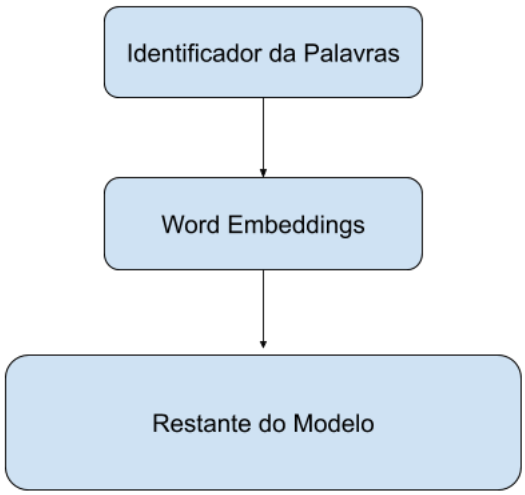

Figura 8.3 – Camadas de Entrada e *Embeddings* do Modelo Simples

Uma das maiores preocupações durante o treinamento de uma RNA, é a escolha de hiper-parâmetros. Esta preocupação está ligada com a capacidade de generalização do modelo, isto é, após o treinamento do modelo, como ele vai desempenhar ao ser alimentado com novos dados para teste. O ideal do modelo é ser capaz de aprender com os dados usados para treinamento, assim evitando o *underfitting*, mas também aplicar o que aprendeu em novos dados, de forma a não ocasionar o *overfitting*. Isto levou aos hiper-parâmetros que podem ser vistos na Tabela 8.2, onde o **número de épocas** combinado ao

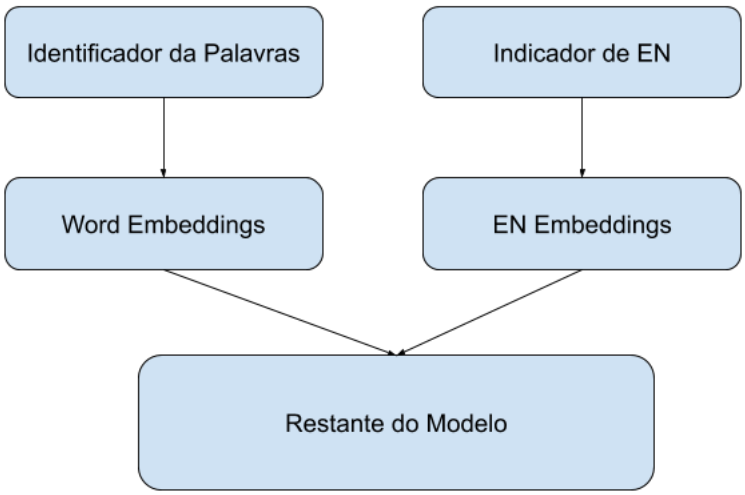

Figura 8.4 – Camadas de Entrada e *Embeddings* do Modelo Médio

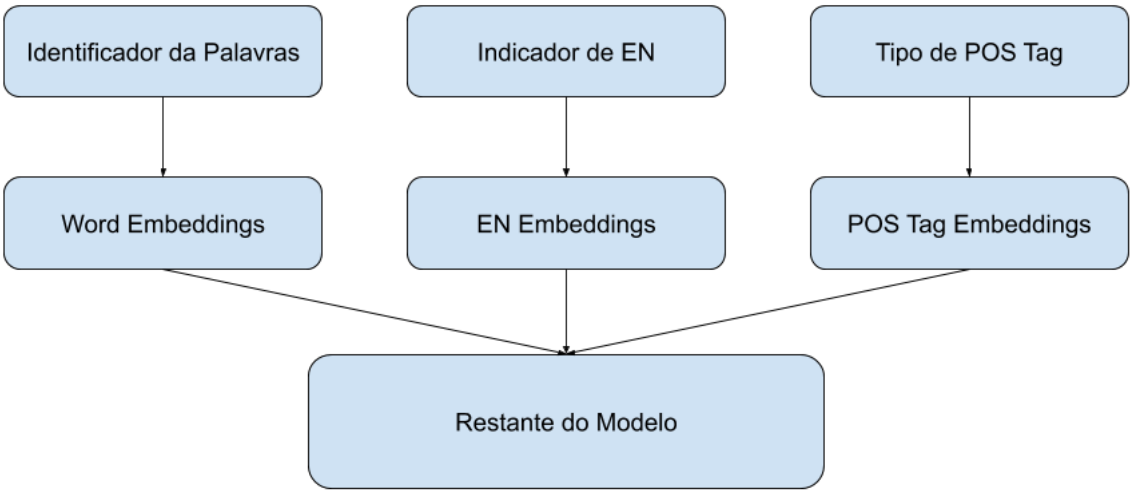

Figura 8.5 – Camadas de Entrada e *Embeddings* do Modelo Completo

tamanho pequeno do ***batch* de treino**, busca amenizar a presença de apenas 90 sentenças no *dataset* de treino, a alta porcentagem de ***Dropout*** junto da baixa porcentagem de **taxa de aprendizado** tenta evitar uma convergência extremamente rápida do modelo, onde ele não conseguiria realmente aprender, e por fim as dimensões de saída de cada camada de *embedding* estão relacionadas com a complexidade de suas respectivas entradas, ou seja, representar a semelhança entre diferentes palavras é muito mais complexo, que os diferentes tipos de POS *Tag*, ou simplesmente valores binários para representar EN ou não EN.

A função de ativação na camada de saída é a função Sigmóide, que tem como característica, o mapeamento de valores de entrada para 0 ou 1, que no caso deste modelo, representam não relacionamento e relacionamento respectivamente. Além disto a função de perda é Entropia Binária Cruzada, que é capaz de calcular a perda com valores binários.

Tabela 8.2 – Hiper-Parâmetros do Modelo Extrator

Hiper-Parâmetro	Valor
Épocas	15
Tamanho das Entradas	100 Palavras
Batch de Treino	3 sentenças
Dropout	50%
Taxa de Aprendizado	1%
Word Embedding	50 dimensões
Algoritmo Word Embedding	Glove
EN Embedding	5 dimensões
POS Tag Embedding	5 dimensões

9. RESULTADOS

Como forma de verificar a performance dos modelos, serão analisados os valores obtidos nas fases de treino e teste. Estes dados serão importantes para compreender melhor o modelo, até onde ele consegue continuar aprendendo sem perder a capacidade de generalização, e se ele realmente consegue realizar a tarefa de ER em sentenças desconhecidas por ele.

9.1 Treino

Os dados presentes nos gráficos de treino, em sua maior parte são provenientes das funções encontradas no *framework Keras*. Entretanto, foi necessário ajustar a função de acurácia, já que o *framework* realiza o cálculo com base em todos os valores de saída do modelo, e como grande parte dela são apenas 0, mesmo que o modelo não acertasse nenhuma predição a acurácia mensurada seria alta. Para tornar o valor mais fiel à realidade,foram comparados apenas as posições em que o modelo fazia a predição de um relacionamento, ou que realmente existisse um relacionamento, este valor encontrado será tratado como **acurácia customizada** deste ponto em diante.

Apesar de todas figuras presentes nesta seção de treino mostrarem gráficos onde o valor de épocas chega sempre em 50, os treinos foram feitos utilizando os hiper-parâmetros apresentados no Capítulo 8. A utilização de mais épocas, serviu apenas para observar o comportamento dos modelos.

As Figuras 9.1 e 9.2 são referentes ao Modelo Simples, este modelo apresenta uma acurácia customizada de no máximo 60%, enquanto o valor de perda vai decaindo rapidamente até a época 20, onde ele fica mais estável até a época final. Seguindo com as Figuras 9.3 e 9.4 temos os gráficos referentes ao Modelo Intermediário, que atinge um pico de aproximadamente 80% de acurácia customizada, e os valores de perda rapidamente começam a convergir. O Modelo Completo é apresentado nos gráficos das Figuras 9.5 e 9.6, que nos mostra dados muito similares ao Modelo Intermediário, tanto em termos de acurácia customizada, quanto nos valores de perda.

Com os gráficos de todos os modelos apresentados, é possível perceber que todos eles se comportam de uma forma muito semelhante, até por volta da época 15, onde eles estão realmente aprendendo, e após isso eles apenas tentam se adequar aos dados de treino, visíveis nas rápidas variações, mas que ficam estagnadas em uma faixa de valores. Este é um dos motivos para a escolha de 15 épocas, que é o momento que ainda não está ocorrendo o *overfitting*.

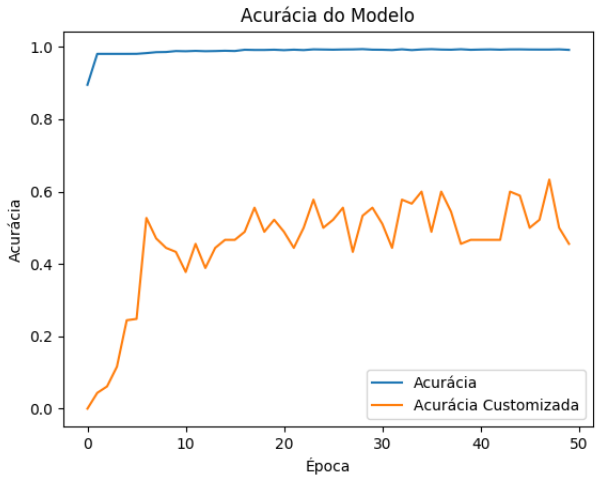

Figura 9.1 – Acurácia do Modelo Simples

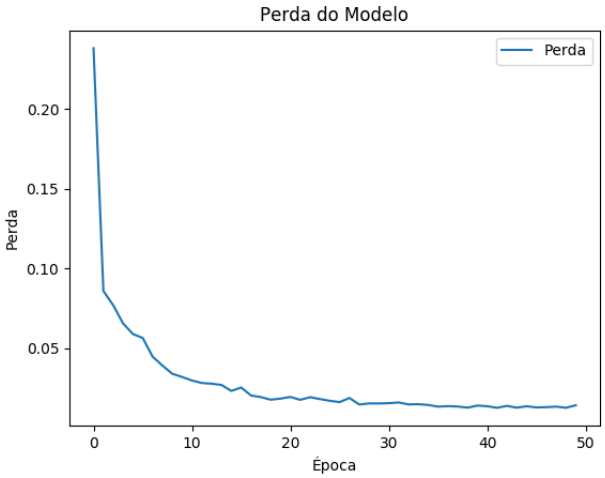

Figura 9.2 – Perda do Modelo Simples

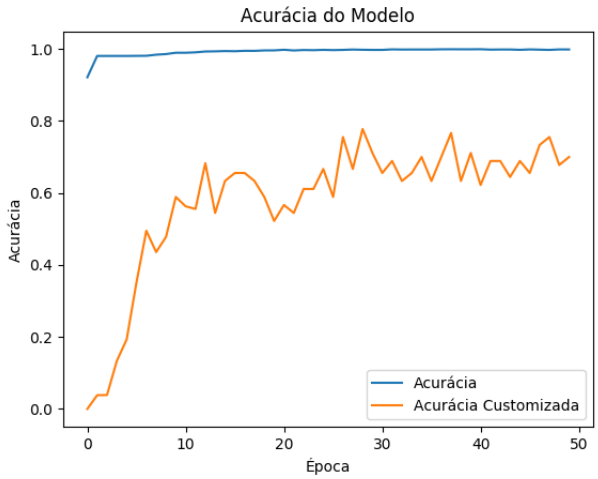

Figura 9.3 – Acurácia do Modelo Intermediário

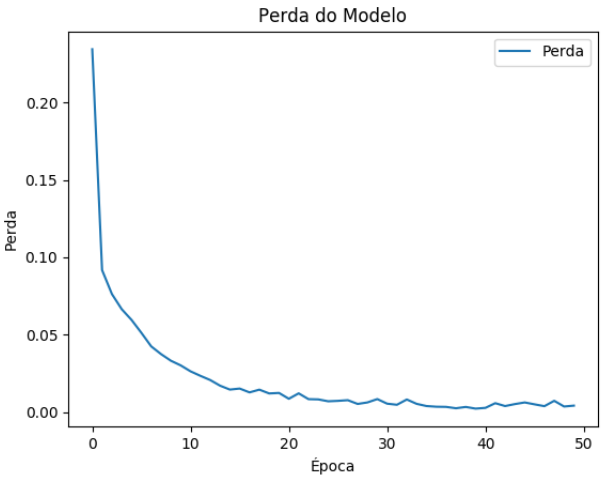

Figura 9.4 – Perda do Modelo Intermediário

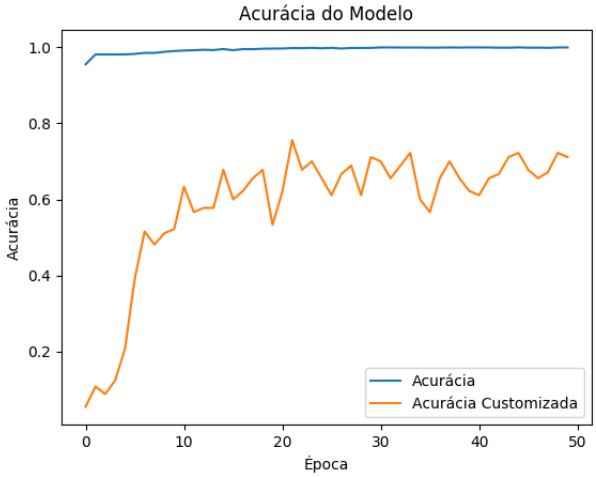

Figura 9.5 – Acurácia do Modelo Completo

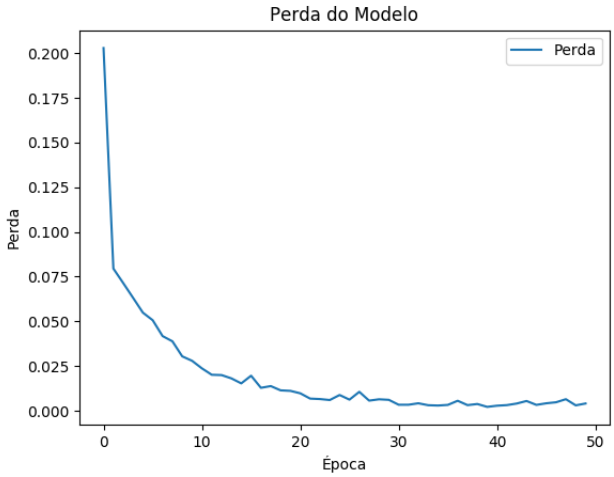

Figura 9.6 – Perda do Modelo Completo

9.2 Teste

9.2.1 Métricas

Para o teste dos modelos, foram fornecidas métricas, junto de fórmulas para estas métricas, por parte dos coordenadores da tarefa do *IberLEF*. Os sistemas submetidos são avaliados quanto a capacidade de extrair tanto relacionamentos completamente corretos, quanto os parcialmente corretos. A avaliação será baseada nos seguintes dados:

- **Relacionamentos Completamente Corretos (RCC)**
- **Relacionamentos Parcialmente Corretos Absolutos (RPCA)**
- **Relacionamentos Parcialmente Corretos (RPC)**
- **Total de Relacionamentos no *dataset* (TR)**
- **Relacionamentos Identificados (RI)**

Cada relacionamento que o sistema acerta completamente é adicionado 1 ao valor de RCC. Os relacionamentos que estão parcialmente corretos seguem o seguinte fluxo: é adicionado 1 ao valor de RPCA, que apesar de não ser utilizado nas métricas propostas pelos organizadores é útil para entender como os modelos estão se comportando, e o valor de RPC para cada relacionamento é calculado seguindo a formula:

$$RPC = 0.5 * \left(\frac{\text{nro. de acertos na predição do relacionamento}}{\max(\text{nro. de palavras no relacionamento, nro. de palavras na predição do sistema})}\right) \tag{9.1}$$

Como métricas principais de avaliação temos:

- **Precisão:** Compara a proporção de respostas corretas, com a proporção de respostas dadas pelo sistema.
- **Recall:** Compara a proporção de respostas corretas, com as respostas esperadas no *dataset*.
- **F-Measure:** Combina as métricas de precisão e *recall*.

Estas métricas foram dividas em duas partes, uma para levar em conta apenas os RCC e outra para incluir os RPC. Desta forma teremos as seguintes formulas para as métricas:

$$\text{Precisão Exata} = \frac{RCC}{RI} \tag{9.2}$$

$$\text{Recall Exata} = \frac{RCC}{TR} \tag{9.3}$$

$$\text{F-Measure Exata} = \frac{2 * \text{Precisão Exata} * \text{Recall Exata}}{\text{Precisão Exata} + \text{Recall Exata}} \tag{9.4}$$

$$\text{Precisão Parcial} = \frac{RCC + RPC}{RI} \tag{9.5}$$

$$\text{Recall Parcial} = \frac{RCC + RPC}{TR} \tag{9.6}$$

$$\text{F-Measure Parcial} = \frac{2 * \text{Precisão Parcial} * \text{Recall Parcial}}{\text{Precisão Parcial} + \text{Recall Parcial}} \tag{9.7}$$

9.2.2 Avaliação dos Modelos

Após a fase de treinamento, e fazendo uso das métricas apresentadas, foram realizadas as avaliações dos modelos desenvolvidos, juntamente do NLPyPort que foi o único sistema submetido ao *IberLEF* e do RelP. Os resultados referentes a cada uma das métricas, de cada um destes modelos podem ser conferidos na Tabela 9.1.

Tabela 9.1 – Resultados dos Modelos

Nome do Modelo	RI	RCC	RPCA	Precisão Exata	Recall Exata	F-Measure Exata	Precisão Parcial	Recall Parcial	F-Measure Parcial
Modelo Simples	133	28	61	0.210	0.187	0.198	0.326	0.288	0.305
Modelo Intermediário	141	75	45	0.531	0.503	0.517	0.588	0.557	0.572
Modelo Completo	132	83	30	0.628	0.557	0.590	0.684	0.606	0.642
RelP	74	46	21	0.621	0.308	0.412	0.685	0.340	0.454
NLPyPort	144	106	—	0.736	0.711	0.723	0.766	0.748	0.757

Após observar as métricas calculadas dos modelos desenvolvidos, juntamente com os outros dois modelos, podemos dizer que o sistema baseado em regras, o NLPyPort teve os melhores resultados. Porém é interessante analisar os resultados obtidos por todos os sistemas, e entender um pouco mais sobre cada um deles.

- **Modelo Simples:** Teve um desempenho ruim, apesar de extrair muitos relacionamentos, aproximadamente 30% deles estavam completamente errados, e o número de RPCA foi aproximadamente o dobro de RCC, mostrando que apenas a *feature* de similaridade de palavras obtida com o *word embedding* não é suficiente para produzir resultados adequados.

- **Modelo Intermediário:** A adição da identificação de EN como entrada no modelo, praticamente dobrou a precisão, tanto a exata como a parcial. Tal ganho de desempenho, deve-se ao fato do modelo ser mais assertivo ao extrair os relacionamentos, de forma que aproximadamente 85% dos relacionamentos estavam corretos de forma completa ou parcial, além do número de RCC ser notoriamente maior que o RPCA.
- **Modelo Completo:** Novamente ao adicionar uma entrada modelo anterior obtivemos ganhos de desempenho, desta vez foi adicionada a entrada de identificador de tipo de POS *Tag*. Apesar do número de RI diminuir quando comparado ao modelo anterior, a soma de RCC e RPCA continuou muito próxima, entretanto o número de RCC passou a ser quase o triplo de RPCA, fazendo com que a precisão aumente 10% quando comparado ao modelo anterior.
- **RelP:** este sistema não teve um número tão grande de RI, aproximando-se apenas de 50% do TR, mas teve uma alta precisão nos relacionamentos que identificou.
- **NLPyPort:** O sistema baseado em regras foi capaz de ter uma grande cobertura de relacionamentos identificados, além de obter valores muito altos em todas as métricas, mostrando o poder de modelos baseados em regras para *datasets* específicos.

Apesar dos resultados apresentados pelos modelos desenvolvidos serem inferiores ao *NPLPyPort*, os Modelos Intermediário e Completo apresentam resultados razoáveis, tendo em vista que eles utilizam RNR e o tamanho do *dataset* da tarefa não é o mais indicado para o treinamento deles, já que normalmente as redes necessitam um grande volume de dados para conseguir realizar um aprendizado capaz de tornar o modelo o mais geral possível. Outro detalhe relevante, que é visível nos resultados apresentados, é a importância da escolha de características que agreguem valor quando usadas como entrada dos modelos, pois sem nenhuma alteração estrutural além das entradas, é notável a evolução de performance dos modelos.

10. CONCLUSÃO

A alta demanda de análise e extração de dados nos tempos atuais, agregou ainda mais valor para as tarefas de PLN, sendo assim, a realização de uma tarefa de PLN presente no *IberLEF* 2019, mostrou-se interessante como um trabalho de conclusão de curso, já que foram explorados muitos conceitos de IA vistos no decorrer do curso. Após a definição de especificamente focar-se na tarefa de ER, foi necessário buscar novos conhecimentos sobre os trabalhos já desenvolvidos na área, e quais abordagens ou até mesmo melhorias poderiam ser utilizadas na busca da solução ao problema proposto.

Dentre as possibilidades levantadas, a escolha de RNR mostrou-se acertada, porque dentre os objetivos propostos para este trabalho, estavam aprofundar conhecimentos em PLN, e ser capaz de analisar resultados e buscar melhores opções ou abordagens caso necessário, pontos que foram alcançados durante o processo de desenvolvimento.

Por fim é possível dizer que os objetivos traçados inicialmente foram plenamente atingidos. Já que foi desenvolvido um sistema capaz de realizar todo o pipeline de pré-processamento de dados necessário para a tarefa, além de ser capaz de realizar o treinamento e teste do modelo, onde apesar de não ter o melhor resultado dos sistemas submetidos, apresenta uma boa performance, e ainda tem ao seu favor a flexibilidade de uma melhor generalização.

10.1 Trabalhos Futuros

Durante o desenvolvimento deste trabalho, foram observados alguns pontos que poderiam ser utilizados em trabalhos futuros. Estes pontos vão de testes com dados diferenciados para observar o comportamento dos modelos desenvolvidos até a modificação de camadas e incremento de complexidade no modelo.

10.1.1 Quantidade de Dados

O aumento do número de dados nos *datasets*, além da incorporação de outros domínios, possibilitaria a verificação da capacidade de generalização dos modelos desenvolvidos. Também possibilitaria um melhor treinamento dos modelos, já que devido ao tamanho reduzido do *dataset* utilizado neste trabalho, não foi possível realizar a separação de uma pequena parte dos dados para validação, como é comum durante a etapa de treino, desta forma poderia ser possível obter melhores resultados.

10.1.2 Customização do Modelo

Neste trabalho foi utilizado o *framework Keras* para a construção da RNR, ele facilitou o processo de implementação dos modelos, mas tirou liberdade em alguns aspectos de customização. A implementação de um modelo desenvolvido de forma mais manual, sem o uso de *frameworks*, poderia trazer maior flexibilidade nas entradas de dados dos modelos, além de um controle muito mais apurado dos processos, e talvez levar a melhores resultados.

Dentro do aspecto de customização, outro ponto a ser explorado em trabalhos futuros são as funções utilizadas na RNR, tanto de perda, como do cálculo de acurácia. Como pôde ser visto nos resultados dos modelos implementados, as funções utilizadas como padrão mostravam dados que não condiziam completamente com a performance dos modelos, já que era feita uma análise total da saída de dados, e não apenas dos relacionamentos que eram o foco do trabalho. A customização trivial da função de verificação de acurácia, trouxe a visualização mais real do desempenho dos modelos, desta forma podemos acreditar que o estudo de novas funções de perda e acurácia mais adequadas ao problema, também seriam capazes de produzir ganhos nos resultados apresentados pelos modelos.

10.1.3 Modelo BiLSTM-CRF

A utilização de modelo BiLSTM-CRF, que é a junção de um modelo BiLSTM com um modelo CRF na camada de saída. Este é um experimento válido visto os bons resultados que este tipo de modelo vem obtendo nas tarefas de REN, como pode ser visto nos números obtidos pelo RelP.

Esta mudança de modelo seria feita através de algumas modificações no Modelo Completo desenvolvido neste trabalho, onde o principal ponto seria a troca da camada de saída, que seria substituída por uma camada utilizando o modelo CRF. A utilização deste modelo híbrido poderia produzir resultados interessantes, visto que um modelo CRF é mais sofisticado e complexo que a função de ativação softmax que é utilizada na camada de saída do modelo atualmente.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] “Colaboratory - frequently asked questions”. Capturado em: https://research.google.com/colaboratory/faq.html, Junho 2019.

[2] “General python faq”. Capturado em: https://docs.python.org/3/faq/general.html#what-is-python, Abril 2019.

[3] “Iberlef 2019 portuguese named entity recognition and relation extraction tasks”. Capturado em: http://www.inf.pucrs.br/linatural/wordpress/iberlef-2019/, Março 2019.

[4] Amaral, D.; Fonseca, E.; Lopes, L.; Vieira, R. “Comparative analysis of portuguese named entities recognition tools”. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14), 2014, pp. 2554–2558.

[5] Augenstein, I.; Das, M.; Riedel, S.; Vikraman, L.; McCallum, A. “SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications”. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017, pp. 546–555.

[6] Burkov, A. “The Hundred-page Machine Learning Book”. Andriy Burkov, 2019.

[7] Cimiano, P. “Ontology Learning and Population from Text: Algorithms, Evaluation and Applications”. Berlin, Heidelberg: Springer-Verlag, 2006.

[8] de Abreu, S. C.; Vieira, R. “Relp: Portuguese open relation extraction”, *Knowledge Organization*, vol. 44–3, 2017, pp. 163–177.

[9] Ferreira, J.; Gonçalo Oliveira, H.; Rodrigues, R. “Nlpyport: Named entity recognition with crf and rule-based relation extraction”. In: Iberian Languages Evaluation Forum (IberLEF 2019), 2019.

[10] Gers, F. “Long short-term memory in recurrent neural networks”, 2001.

[11] Goodfellow, I.; Bengio, Y.; Courville, A. “Deep Learning”. MIT Press, 2016, http://www.deeplearningbook.org.

[12] Hartmann, N.; Fonseca, E. R.; Shulby, C.; Treviso, M. V.; Rodrigues, J.; Aluísio, S. M. “Portuguese word embeddings: Evaluating on word analogies and natural language tasks”, *CoRR*, vol. abs/1708.06025, 2017, 1708.06025.

[13] Haykin, S. S. “Neural networks and learning machines”. Upper Saddle River, NJ: Pearson Education, 2009, third ed..

[14] Hearst, M. A. “Automatic acquisition of hyponyms from large text corpora”. In: Proceedings of the 14th Conference on Computational Linguistics - Volume 2, 1992, pp. 539–545.

[15] Hendrickx, I.; Kim, S. N.; Kozareva, Z.; Nakov, P.; Ó Séaghdha, D.; 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 Workshop on Semantic Evaluations: Recent Achievements and Future Directions, 2009, pp. 94–99.

[16] Hochreiter, S.; Schmidhuber, J. “Long short-term memory”, *Neural Comput.*, vol. 9–8, Nov 1997, pp. 1735–1780.

[17] Jurafsky, D.; Martin, J. H. “Speech and Language Processing (2nd Edition)”. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009.

[18] Kriesel, D. “A Brief Introduction to Neural Networks”. 2007.

[19] Lee, J. Y.; Dernoncourt, F.; Szolovits, P. “MIT at SemEval-2017 task 10: Relation extraction with convolutional neural networks”. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017, pp. 978–984.

[20] Li, Y.; Xu, L.; Tian, F.; Jiang, L.; Zhong, X.; Chen, E. “Word embedding revisited: A new representation learning and explicit matrix factorization perspective”. In: Proceedings of the 24th International Conference on Artificial Intelligence, 2015, pp. 3650–3656.

[21] Santos, D.; Freitas, C.; Gonçalo Oliveira, H.; Carvalho, P. “Second harem: New challenges and old wisdom”, 2008, pp. 212–215.

[22] Sarawagi, S. “Information extraction”, *Found. Trends databases*, vol. 1–3, Mar 2008, pp. 261–377.

[23] Singh, S. “Natural language processing for information extraction”, *CoRR*, vol. abs/1807.02383, 2018, 1807.02383.

[24] Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; Zhao, J. “Relation classification via convolutional deep neural network”, *the 25th International Conference on Computational Linguistics: Technical Papers*, 01 2014, pp. 2335–2344.

[25] Zhang, D.; Wang, D. “Relation classification via recurrent neural network”, *CoRR*, vol. abs/1508.01006, 2015.