CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Thiago Felipe Schuch

thigschuch@rede.ulbra.br

REDE NEURAL PARA IDENTIFICAÇÃO DE INTERESSADOS E INVESTIGADOS EM PROCEDIMENTOS EXTRAJUDICIAIS

Palmas – TO

2021

Thiago Felipe Schuch

REDE NEURAL PARA IDENTIFICAÇÃO DE INTERESSADOS E INVESTIGADOS EM PROCEDIMENTOS EXTRAJUDICIAIS

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Araújo Castro.

Palmas – TO

2021

Thiago Felipe Schuch

REDE NEURAL PARA IDENTIFICAÇÃO DE INTERESSADOS E INVESTIGADOS EM PROCEDIMENTOS EXTRAJUDICIAIS

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Araújo Castro.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Esp. Fábio Araújo Castro

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Me. Jackson Gomes de Souza

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Fernanda Pereira Gomes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2021

“O mundo é um lugar perigoso, não por causa daqueles que fazem o mal, mas por causa daqueles que olham e não fazem nada.”
(Mr. Robot)

RESUMO

SCHUCH, Thiago Felipe. **Rede neural para identificação de interessados e investigados em procedimentos extrajudiciais**. 2021. 26 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2021[1].

O âmbito extrajudicial consiste em casos leves em que não é necessário intervenção da Justiça, por esse motivo, diversos procedimentos são iniciados diariamente, levando a uma tarefa cansativa e repetitiva: a identificação de pessoas citadas em documentos. Entretanto, existem tecnologias que podem ser utilizadas para diminuir ou sanar esta tarefa. O reconhecimento de entidades nomeadas pode ser trabalhado com base em redes neurais e em processamento de linguagem natural para assim entregar informações precisas do que foi extraído do texto. Essa combinação de tecnologias é capaz de ler, interpretar e retirar elementos de documentos, utilizada dessa forma neste trabalho para ter a habilidade de identificar quem são as pessoas investigadas e quem são os interessados dentro de cada procedimento extrajudicial. Logo, o presente trabalho tem o intuito de automatizar essa tarefa repetitiva e cansativa de leitura e identificação de pessoas.

PALAVRAS-CHAVE: Procedimentos Extrajudiciais, Processamento de Linguagem Natural, Reconhecimento de Entidades Nomeadas, Redes Neurais.

[1] Elemento incluído com a finalidade de posterior publicação do resumo na internet. Sua formatação segue a norma ABNT NBR 6023, por isto o alinhamento e o espaçamento diferem do padrão do texto.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Bi-LSTM	- Bi-*Long Short-Term Memory*
CSV	- *Comma Separated Values*
GPU	- *Graphics Processing Unit*
HTML	- *Hypertext Markup Language*
IA	- Inteligência Artificial
JSON	- *JavaScript Object Notation*
LSTM	- *Long Short-Term Memory*
LNTK	- *Natural Language Toolkit*
PLN	- Processamento de Linguagem Natural
RAM	- *Random-Access Memory*
REN	- Reconhecimento de Entidades Nomeadas
ReLU	- *Rectified Linear Unit*
RNA	- Rede Neural Artificial
RNR	- Rede Neural Recorrente
TPU	- *Tensor Processing Unit*
WE	- *Word Embedding*

SUMÁRIO

1 INTRODUÇÃO

O Direito possui o aspecto de ser justo, de manter a segurança da sociedade e de desenvolver bem-estar econômico, social e cultural (DUARTE, 2017). Por isso mesmo, ao longo dos séculos, ocasionou a criação de vários tipos de documentos judiciais para se ter registro de fatos. Atualmente existem documentos para todas as ocasiões, desde petições que são usadas para requerer direitos de um indivíduo até termos de uso e políticas de privacidade, bastante comuns no ambiente online. Essa quantidade de tipos de documentos gera um número expressivo de procedimentos. Como exemplo, o Relatório Justiça em Números 2020 apresenta que ainda existiam 77 milhões de procedimentos em tramitação no final do ano de 2019 (CNJ, 2020).

O início de um procedimento judicial se dá quando alguém registra uma denúncia, seja ela diretamente ao promotor ou pelo *website* do Ministério Público. O processo então tem várias etapas, muitas delas envolvendo cartórios. Como Pinto (2007, p. 11) explica, "cartórios são como Serviços Auxiliares de Justiça, funcionando no âmbito extrajudicial, sem interferência direta do Estado através do Poder Judiciário". Nos cartórios os procedimentos recebem o tipo adequado, o assunto e área do qual se tratam e se possível, interessados e investigados, todos de forma manual.

Além das etapas que os procedimentos passam e do grande número de documentos gerados, é exigido pessoas capazes de lerem e identificarem certos pontos objetivos dentro desses textos, contudo, máquinas já conseguem realizar esse trabalho (SILVA, 2021). A identificação antecipada de informações fornece agilidade no procedimento, entretanto, essa identificação exige ler e analisar documento por documento tentando extrair os dados que melhor preencherão os campos requisitados. Essa leitura torna-se uma tarefa repetitiva, cansativa e complicada devido à quantidade de procedimentos -- como exemplo, o Ministério Público do Tocantins teve cerca de 16 mil procedimentos nos últimos dois anos. A identificação geralmente demanda uma pessoa específica que tem conhecimento na área e que ainda consome bastante de seu tempo.

O Processamento de Linguagem Natural (PLN) é uma área da Inteligência Artificial (IA) capaz de ler e extrair informações de textos, realizando até mesmo processamento léxico e sintático da linguagem (SHOWDHURY, 2003). O avanço da área de processamento natural fez com que várias ferramentas fossem desenvolvidas para trabalhar e automatizar tarefas voltadas à linguística, uma dessas ferramentas é o NLTK (*Natural Language Toolkit*).

A leitura e extração de dados utilizando PLN é fundamental para este trabalho, mas também se faz necessário identificar os tipos desses objetos extraídos de dentro do texto,

utilizando assim o Reconhecimento de Entidades Nomeadas (REN). O REN é capaz de analisar texto de forma livre e identificar as ocorrências de entidades, classificando-as como pessoas, localizações, organizações, tempo, entre outros (MARRERO et al., 2013).

Para aumentar a taxa de confiabilidade do REN, pode-se mesclar tecnologias como Inteligência Artificial, Aprendizado de Máquina e/ou Redes Neurais e derivados, a exemplo do que é proposto neste trabalho. As redes neurais são algoritmos que possuem conexões entre si e que, juntos, simulam os neurônios biológicos, sendo necessário algum tipo de treinamento para que a rede compreenda o resultado esperado (ALVAREZ & LUQUE, 2003).

Posto isto, esses tipos de documentos para se tornarem completos e terem um destino definido, precisa-se ler e extrair informações manualmente. Apoiando-se no âmbito extrajudicial, o uso da REN possibilita ler e extrair as principais informações do texto, neste caso, buscando saber das pessoas envolvidas. Já para aumentar a acurácia dessa informação e predizer quem é investigado e quem é interessado, uma rede neural do tipo *Long Short Term Memory* (LSTM) é responsável por esta tarefa.

Por fim, o desenvolvimento de um algoritmo baseado em redes neurais possivelmente transformará esta tarefa de leitura e interpretação em uma rotina que levará menos tempo para ser realizada e sem qualquer interferência humana. Se a rede tiver um grau de precisão elevado, talvez possibilitará a realocação de colaboradores a outras tarefas da instituição.

2 REFERENCIAL TEÓRICO

2.1 Documentos Judiciais Digitais

A lentidão do Estado no Brasil foi o motivo pelo qual desencadeou-se a adoção e desenvolvimento de várias tecnologias sobre o processo jurídico brasileiro. Essa mudança começou no ano de 1999 através de fax, seguido por um artigo em 2001 da Justiça Federal autorizando intimações e petições *online*. Deu-se então o desenvolvimento do sistema E-Proc em 2004, que foi e é bastante importante até hoje (PEREIRA, 2011).

O E-Proc foi projetado especificamente para trazer mais rapidez e comodidade ao sistema jurídico brasileiro. Segundo Rabelo et al. Garcia (2009, p. 11-12), "o sistema pretendia facilitar o trabalho de advogados e procuradores, bem como melhorar a qualidade de atendimento às partes e agilização e segurança no trâmite de processos e atuações". O sistema foi tão bem aceito no âmbito judicial brasileiro que em 2016 atingiu 4 milhões de procedimentos virtuais (PEIXOTO, 2016).

Apesar do avanço digital e de sistemas, foi somente a partir do ano de 2006 que os processos poderiam ser realizados, julgados e reproduzidos na Internet. Nesse ano, aconteceu a publicação da Lei Nº11.419-2006, sendo um marco para o processo jurídico eletrônico brasileiro. A partir dessa lei foi autorizado o uso da Internet para tramitação e acompanhamento de procedimentos. Foi também permitido a criação e uso de sistemas no processamento das ações. Ela além disso, aprovou a comunicação pela Internet e até o início do uso de assinaturas eletrônicas (BRASIL, 2006).

Além dos procedimentos comuns, existem os procedimentos extrajudiciais. Pelo significado da palavra extrajudicial, eles são ações que são resolvidas sem intervenção da Justiça, geralmente intermediadas por um advogado e resolvidas de forma conciliatória ou amigável (VELOSO et al., 2016). Registrados pelo Ministério Público do Tocantins, esse tipo de procedimento soma 32 mil itens públicos desde o ano de 2016 até este momento, em 2021. Sendo que 11 mil procedimentos ainda estão ativos, ocasionados tanto pela quantidade excessiva registrada quanto pela complicação em certas etapas dos processos. Essa demora para seguimento dos processos se dá pois eles ficam aguardando pessoas responsáveis para ler e decidir o seu próximo destino.

2.2 Redes Neurais Artificiais (RNA)

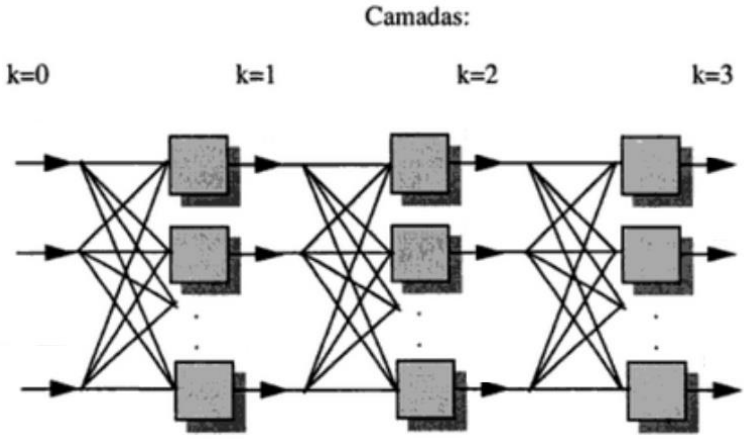
A ideia das redes neurais artificiais (RNAs) teve sua inspiração nos neurônios biológicos, adaptando suas características e comportamentos para o ambiente virtual. Como Matsunaga (2012) diz, uma rede neural conta com diversos neurônios artificiais que

possibilitam coletar, utilizar e armazenar informações baseadas em treinamento. Além da habilidade de simular o neurônio biológico, essas redes apresentam robustez, tolerância a falhas, flexibilidade e processamento de informações incompletas.

Segundo Kovács (2006), McCulloch publicou em 1943 o primeiro artigo sobre RNAs, contendo uma rede de apenas alguns neurônios chamada de discriminador linear, essa rede era capaz de implementar várias funções booleanas. Apesar de simples, a rede de McCulloch foi bastante inovadora para a época. Até que em 1960, Roseblatt, propôs uma série de experimentos chamados de *perceptrons*.

Perceptron é uma rede neural com vários neurônios discriminadores lineares, sua principal característica é a quantidade de camadas que possui (ROSENBLATT, 1960). As camadas de uma rede neural são constituídas de *n* neurônios e cada neurônio realiza as funções que está destinado a fazer. Após isso, dependendo da implementação, pode enviar a sua informação de saída para outro neurônio que está na camada seguinte ou para todos os neurônios seguintes.

Figura 1: Representação de uma Rede Neural Multicamadas



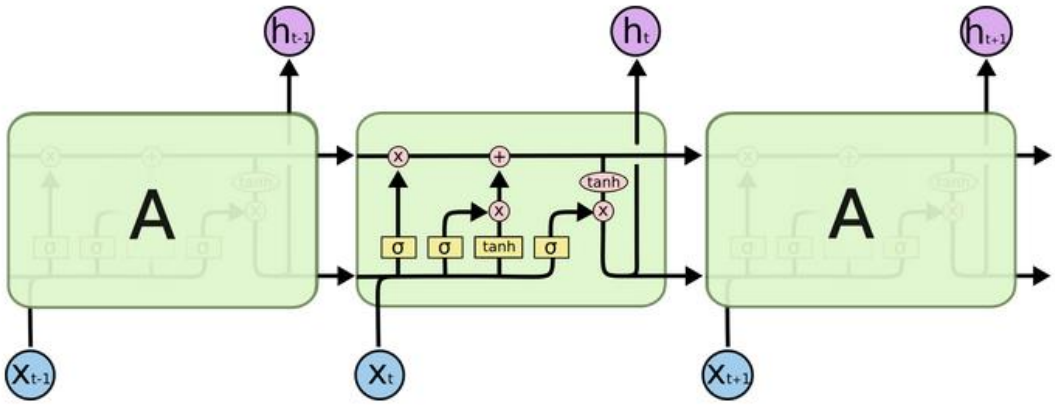
Fonte: KOVÁCS, 2006

A figura 1 exemplifica uma rede neural. As camadas são representadas pelas letras “k”, os neurônios por quadrados e as linhas simbolizam os pesos que a informação atualmente dentro do neurônio possui. Todas as camadas recebem informações para que cada neurônio a processe. Essas informações podem ser fornecidas pelo indivíduo humano, como acontece na camada k0, chamada camada de entrada, ou pela própria rede, no caso das camadas internas k1 e k2, que recebem a informação passada pelas camadas anteriores. A k3 é a última camada, descrita como camada de saída, as informações que chegaram pela camada de entrada sairão devidamente processadas pela k3. Em suma, a camada de entrada é por onde os dados iniciais são inseridos,

nas camadas internas ou ocultas ocorre os cálculos e processamento, por fim a camada de saída finaliza o processo apresentando o resultado.

Ao longo dos anos, mais tipos de redes neurais foram sendo desenvolvidas, assim como as redes neurais *Long Short Term Memory* (LSTM) ou também chamadas de Redes Neurais Recorrentes (RNR). Elas possuem esse nome devido a sua capacidade de esquecer ou de lembrar de determinado evento dentro de suas camadas internas. Para realizarem esse tipo de controle da memória, a cada saída de uma camada interna, podem haver funções de ativação que determinam se alguma das informações processadas naquela camada será esquecida ou lembrada, ou seja, ela poderá ser descartada ou guardada dentro da rede para ser usada posteriormente (JULIANI, 2019).

Figura 2: Representação de uma Rede Neural LSTM



Fonte: OLAH, 2015

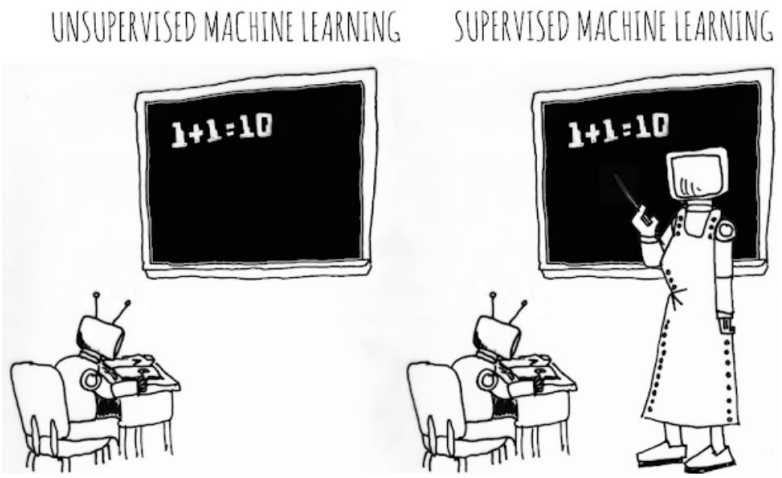
Na figura 2 pode-se observar três retângulos verdes que se repetem, indicando uma estrutura iterativa de nós de uma rede LSTM, assim como as funções de ativação nos chamados portões (*gates*), representados por elipses rosas. Representada pela letra grega *sigma* (σ) está a função *sigmoid*, que aponta números entre 0 e 1 para decidir o quanto da informação recebida pode continuar seu caminho pela rede (OLAH, 2015).

Para que uma rede neural decida qual informação ela deverá fornecer de um determinado conjunto de dados, ela precisa ser treinada e ensinada sobre qual é o seu objetivo. Esse treinamento faz parte de uma subárea da IA chamada de Aprendizado de Máquina. Como Batista (2003, p.11 e 12) diz “a capacidade de aprender é essencial para um comportamento inteligente”, assim a rede neural deste trabalho precisa ser inteligente o bastante para distinguir pessoas de lugares e saber reconhecer o papel dessas pessoas nos documentos extrajudiciais.

O aprendizado de máquina pode ser dividido em três tipos: supervisionado, não supervisionado e por reforço. O ensino supervisionado remete-se à condução do desenvolvedor da rede neural sobre o que é certo ou errado, geralmente utilizando um conjunto de dados para

que a rede interprete casos reais e saídas desejadas. O ensino não supervisionado ocorre quando a rede deve sozinha interpretar se as suas saídas estão certas ou erradas. Já o ensino por reforço remete-se a punição ou gratificação da rede para escolhas corretas ou erradas (SIMÕES, 2006).

Figura 3: Charge sobre aprendizados supervisionados e não supervisionados



Fonte: PROOFF READERS WHIMSY, 2014

A charge da Figura 3 faz alusão ao ensino com e sem professor (com e sem supervisão), onde o aluno, no primeiro quadro deve tentar aprender como realizar a operação matemática de adição sem a ajuda do professor, já no segundo, ele conta com o apoio ou supervisão do professor.

2.3 Processamento de Linguagem Natural (PLN)

Atualmente existem mais de 7 mil línguas (EBERHARD *et al*., 2021) e todas possuem um aspecto em comum, devido a isso, são necessários três tipos de análises para compreendê-las corretamente:

- Léxica - os símbolos presentes nas palavras devem fazer parte do alfabeto da língua;
- Sintática - as palavras devem estar na ordem correta;
- e Semântica - as sentenças precisam expressar um significado.

Exemplos a respeito da análise léxica são: não se pode escrever na língua inglesa com as letras “ç” ou “ã”, esse tipo de acentuação não faz parte do alfabeto inglês, assim como a língua russa utiliza o alfabeto cirílico em vez do alfabeto latino (HENRIQUES, 2018).

Para o texto ser compreendido pelas máquinas, o processo de entendimento da língua também acontece de forma semelhante na computação. Ela Kumar (2010) descreve o processamento de linguagem natural em 6 análises, ordenadas em: fonológica, morfológica, léxica, sintática, semântica, pragmática e discursiva. A análise fonológica é aplicada se for utilizada a fala. A análise morfológica separa as palavras em morfemas, isto significa que as

palavras são quebradas em blocos básicos da gramática, como por exemplo a palavra "Localizar" que tem como base "local" e o sufixo "izar". A análise pragmática e discursiva consiste na relação da linguagem e no contexto usado, as sentenças devem ter coerência entre si, não devem ser simplesmente jogadas no texto.

Um dos principais procedimentos no estudo de Linguagem Natural é o *Word Embedding* (WE, Incorporação de Palavras), este procedimento consiste em vetorizar a representação de palavras incorporando seus significados semânticos e sintáticos recebidos de um *corpus* (WANG, 2019). O *corpus* ou *dataset* vem da Linguística de Corpus e é um conjunto de palavras escolhidas precisamente da língua retratada para um posterior tratamento ou embasamento (SARDINHA, 2004).

Os *Word Embeddings* podem ser divididos em arquiteturas hierárquicas ou em incorporação contextualizada, o primeiro trabalha diretamente na palavra utilizando diversos algoritmos de incorporação de palavras juntamente com redes neurais. Já o segundo, também trabalhando com redes neurais, tenta entender diferentes significados para a mesma palavra dependendo de seu contexto (AKBIK, 2019), como João Pessoa, pode ser tanto o nome de alguém quanto o nome de uma cidade.

Um dos WE que buscam a contextualização é o *Flair Embedding*, proposto em um artigo de Arkib *et. al.* em 2018, ele aplica uma arquitetura de nomeação sequenciada a uma rede neural Bi-LSTM e empilha os diferentes significados encontrados. Esse modelo proposto alcançou o novo estado da arte para o REN na língua inglesa e o aumentou em outras línguas (AKBIK, 2018).

O artigo de 2019 de Wagner *et. al.* apresenta o uso do Google *Natural Language Processing* na extração de textos de normas jurídicas. Em seu trabalho, utilizam diversos filtros para a extração de informações e também um organizador de sentenças chamado LexML para que no final sejam apresentados os resultados em forma de "definição" e "significado" conforme o conjunto de dados pré-processados por eles (TEIXEIRA *et. al.*, 2019).

2.4 Reconhecimento de Entidades Nomeadas (REN)

Enfim, utilizando o processamento de linguagem natural visto anteriormente, a técnica de identificar objetos reais e atribuir determinadas classificações às palavras de um texto é chamada de reconhecimento de entidades nomeadas (MIKHEEV *et. al.*, 1999). Os tipos mais comuns de nomeações que podem ser identificados em um texto livre são locais (LOC), pessoas (PER) e organizações (ORG).

Figura 4: Tipos de nomeações

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridge, etc.
ORG	Companies, agencies, institutions etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.

Fonte: GEEKS FOR GEEKS, 2021

Como a Figura 4 aponta, existem diversos outros tipos de nomeação, cada contexto pode ter seu próprio tipo de nomeação e o mais adequado para a situação. Por exemplo, a frase “Thiago fala inglês e estuda na Ulbra” tem três objetos que poderiam ser identificados: Thiago como pessoa, inglês como linguagem e Ulbra como organização.

Essa identificação necessita um *dataset*, que como visto anteriormente, são textos classificados de forma que, posteriormente, seja possível trabalhar baseado neles.

Figura 5: Texto nomeado

Otávio	B-PESSOA
Portes	I-PESSOA
,	O
16ª	B-ORGANIZACAO
CÂMARA	I-ORGANIZACAO
CÍVEL	I-ORGANIZACAO
,	O
julgamento	O
em	O
28/09/2017	B-TEMPO

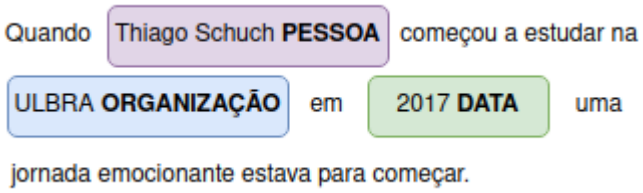
Fonte: ARAÚJO *et. al.*, 2018

O exemplo ilustrado na Figura 5 mostra alguns tipos de nomeações, como Otávio Portes como PESSOA, a 16ª Câmara Cível como uma ORGANIZAÇÃO e 28/09/2017 como TEMPO. Estes exemplos foram retirados da pesquisa de Araújo *et al.* (2018) no qual propunha-se um *dataset* de entidades nomeadas em textos legais brasileiros utilizando redes de memória de longo prazo. O projeto envolveu 7.827 mil sentenças para treinamento, 1.176 sentenças para desenvolvimento e outras 1.389 para teste da rede neural.

Existem também ferramentas como o *SpaCy*, uma biblioteca em *Python* que faz o processamento de nomeação de palavras por meio do PLN. Em sua documentação constam as

habilidades de *tokenização*, classificação de texto, treino de modelos, serialização, nomeação de entidades e outros (SPACY, 2021).

Figura 6: Visualizador *Spacy*



Fonte: Adaptado de SPACY, 2021

Na Figura 6 mostra um exemplo de como é trabalhar com os dados do *SpaCy*, pode-se notar três itens destacados: “Thiago”, “ULBRA” e “2016”. Esses itens são chamados de entidades por fazerem parte de um grupo de classificação, no caso “Pessoa”, “Organização” e “Data”, respectivamente.

3 MATERIAIS E MÉTODOS

3.1 MATERIAIS

As tecnologias que foram utilizadas no desenvolvimento deste trabalho envolveram, principalmente, a linguagem de programação *Python*, visto que ela possui diversas bibliotecas e ferramentas para o desenvolvimento de aplicações da área de Inteligência Artificial (IA). *Keras* e o *Tensorflow* são dois exemplos de bibliotecas bastante populares voltadas para a área de IA.

Foi empregado o uso de bibliotecas nativas do *Python* como JSON, CSV e RE para, respectivamente, exportar textos organizados, ler arquivos de texto separados por vírgula e remover caracteres desnecessários dos textos.

Além de usar a RE para remoção de caracteres desnecessários, foi utilizado também o módulo de *stopwords* do NLTK para a mesma finalidade, pois esse módulo trás palavras que não agregam sentido às frases e assim pode-se realizar um melhor tratamento no texto.

A biblioteca *BeautifulSoup* consegue analisar textos em HTML. Ela serviu para a leitura de pontos específicos dos textos em CSV devido estarem organizados em formato HTML e facilitar a navegação dentro das *tags* do documento.

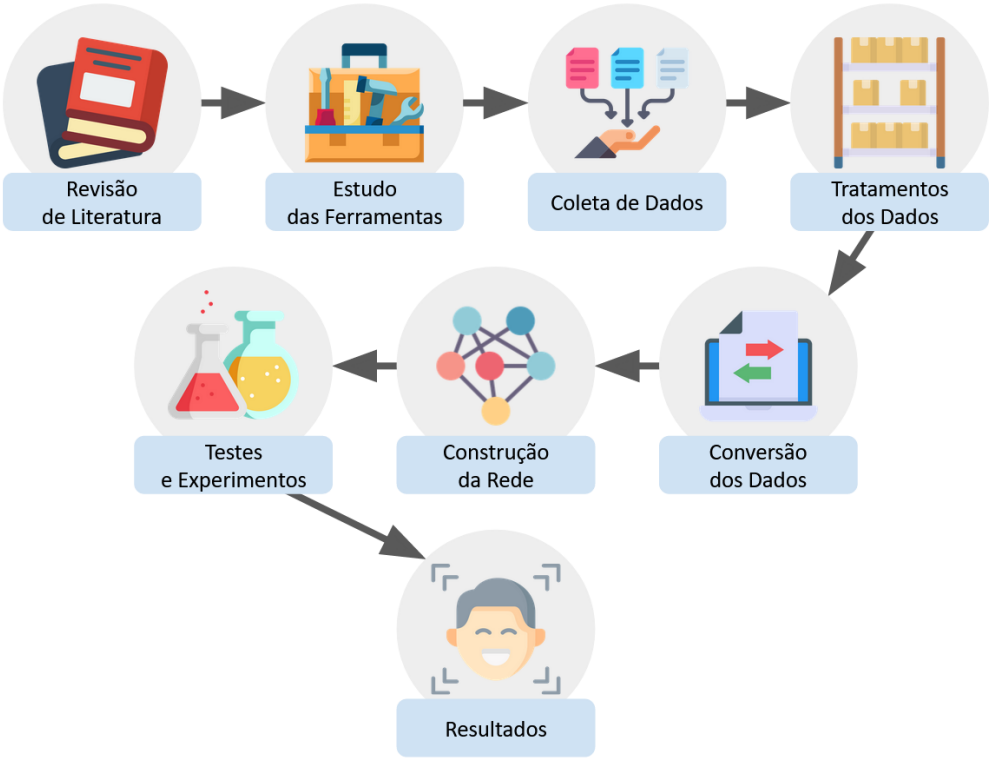
Para facilitar a identificação de nomes de pessoas e organizações dentro dos textos trabalhados, foi utilizada a biblioteca *SpaCy* no *Python*, ela possui modelos prontos para diversos idiomas. Assim, o trabalho de identificar quem é interessado ou investigado dentro do texto fica apenas com a rede neural desenvolvida.

O treinamento de uma rede neural acaba exigindo bastante *hardware* devido a realização de diversos cálculos a respeito dos dados entre os nós da rede. O Google oferece um ambiente virtualizado chamado Google *Collaboratory* (*Colab*) onde é possível obter uma máquina com recursos dedicados a algoritmos dessa natureza. O uso desse ambiente permitiu desenvolvimento e resultados mais rápidos sendo que além dos recursos dedicados, não foi necessário instalar e configurar as bibliotecas do *Keras* e *Tensorflow*.

3.2 MÉTODOS

O desenvolvimento deste trabalho foi realizado em etapas conforme ilustrado na Figura 7.

Figura 7: Etapas do desenvolvimento



Fonte: Autoria própria

O primeiro passo foi realizar a revisão de literatura de trabalhos que ajudassem a entender sobre o ambiente extrajudicial e sobre as redes neurais. Utilizando principalmente o Google Scholar como base de pesquisa.

O segundo passo foi estudar as ferramentas que possivelmente seriam utilizadas no trabalho, escolher qual linguagem de programação e quais bibliotecas seriam mais apropriadas para a realização do projeto.

O terceiro passo consistiu na coleta de dados e para isso um especialista do domínio do Ministério Público disponibilizou um arquivo em formato .CSV (valores separados por vírgula) contendo três colunas: “Interessado”, “Investigado” e “Texto”.

Figura 8: Exemplo do Arquivo .CSV recebido

Interessado	Investigado	Texto
SECRETARIA ESTADUAL DA SEGURA	MAJOR NEGREIROS	MIGUEL BATISTA DE SIQUEIRA FILHO</ 22ª Promotoria De Justiça Da Capital
A COLETIVIDADE	INTENSICARE UTI IOP LTDA.	FRANCISCO JOSE PINHEIRO BRANDES Promotoria De Justiça De Cristalandia
TRIBUNAL DE JUSTIÇA DO ESTADO D	PREFEITO MUNICIPAL DE CARM	FRANCISCO JOSE PINHEIRO BRANDES Promotoria De Justiça De Cristalandia
MUNICIPIO DE LAGOA DA CONFUSÃO	PREFEITO MUNICIPAL DE CARM	FRANCISCO JOSE PINHEIRO BRANDES Promotoria De Justiça De Cristalandia
ANONIMO	SUELENE LUSTOSA MATOS	FRANCISCO JOSE PINHEIRO BRANDES Promotoria De Justiça De Cristalandia

Fonte: Autoria própria

A Figura 8 exemplifica como está estruturado o arquivo recebido, havendo na coluna "Texto" um código HTML das principais informações sobre o respectivo processo entre o investigado e interessado.

No quarto passo houve o tratamento dos dados, no caso, com foco no texto escrito em HTML para extrair a fala de quem redigiu o texto e assim utilizá-lo na rede neural desenvolvida posteriormente. Para trabalhar com o texto HTML foi utilizada a biblioteca *BeautifulSoup* que converte o texto HTML em objetos da linguagem de programação e assim foi possível coletar determinadas informações dentro do texto.

O texto extraído ainda possuía as chamadas *stopwords*, palavras que não acrescentam tanto significado às frases e que podem ser removidas sem afetar o sentido do texto. Para essa remoção, além de um dicionário próprio de palavras, foi utilizado o módulo *stopwords* da biblioteca NLKT e um conjunto de expressões regulares para a remoção de caracteres especiais e outros elementos. Com os dados devidamente tratados, para selecionar quais dados iriam para a rede neural foi feita a identificação das entidades (pessoas e organizações) de cada texto com a biblioteca *SpaCy* e coletadas as cinco palavras anteriores e posteriores a cada entidade.

O quinto passo consistiu na conversão do texto captado no quarto passo para uma forma legível computacionalmente, ou seja, as máquinas processam zeros e uns, ativos e inativos, e assim cada texto foi convertido e salvo utilizando o método *One-Hot Encoding*, onde para cada palavra do *dataset* existe um número, e se ele existe no trecho analisado, este número será um, caso contrário, será zero.

$$[nesta, desta, justiça, público, \dots] = [0,1,1,0, \dots]$$

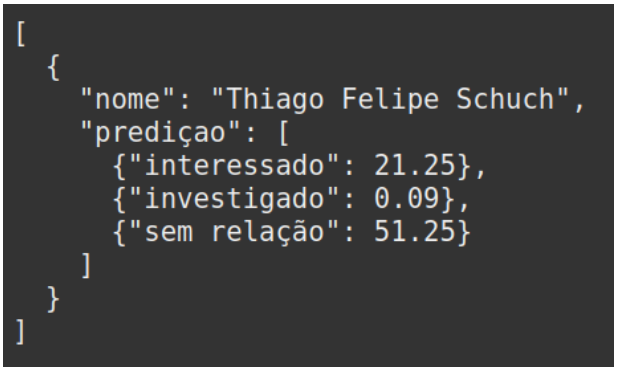
No exemplo, as palavras que precedem a igualdade são todos os cinco termos anteriores e posteriores às entidades e após a igualdade existe um elemento binário que equivale a essas palavras, sendo o numeral zero para “palavra inexistente no texto” e o numeral um para “palavra existente no texto”. O exemplo então trás que o texto analisado possuía as palavras “desta” e “justiça” antes e/ou depois da entidade encontrada. Com essa lista de zeros e uns, foi feita a categorização manual de cada entidade, atribuindo a ela um número entre 0 e 2, sendo que 0, a entidade em questão não tem relação com o texto, sendo 1 ela é investigada e se 2, ela é da parte interessada. Essa categorização e o texto convertido formam o *dataset* que conteve 408 entidades e 1174 palavras, sendo utilizado no desenvolvimento e que também serviu para treino e validação da rede neural.

No sexto passo foi implementada a rede neural utilizando *Keras* e rodando sobre o *TensorFlow* dentro do ambiente do Google *Collaboratory*, e os dados convertidos no quinto passo serviram de entrada para a rede como fins de treinamento. A rede neural do tipo *Long Short Term Memory* foi escolhida para o projeto por possuir “memória”, ela precisa saber o que já foi passado ou não pela rede e também ter a capacidade de esquecer certos elementos para assim tentar prever qual será o próximo elemento.

O sétimo passo consistiu em realizar experimentos com as camadas e parâmetros da rede, manipulando as gerações de cada processamento, adicionando e removendo diferentes tipos de camadas, analisando os gráficos de perda e acurácia gerados, bem como o conjunto de validação. Os dados de validação foram extraídos dos dados totais, uma quantia de 20%, deixando os outros 80% para treino.

Por fim, o oitavo passo possibilitou exibir a saída da rede, exibindo ao usuário um campo de entrada em que ele pode digitar ou colar um texto e a rede neural tentará fazer a predição e retornará o que ela conseguiu extrair.

Figura 9: Saída da Rede Neural



Fonte: Autoria própria

Na figura 9 consta um exemplo de saída a rede, onde é um objeto JSON contendo em sua chave o nome da entidade reconhecida e nos valores, um conjunto com três números flutuantes a respeito das chances dessa entidade ser “Nada no texto”, “Investigada” ou “Interessada”, respectivamente.

4 RESULTADOS E DISCUSSÃO

Esta seção apresenta em detalhes como o projeto da rede neural foi desenvolvido. É demonstrado as etapas de coleta de dados e seu tratamento, a conversão do texto em código de máquina, a construção da rede, seus testes e análises e sua saída sendo exibida aos usuários.

4.1 COLETA E TRATAMENTO DOS DADOS

O Ministério Público do Tocantins em seu Portal do Cidadão (*https://mpto.mp.br/cidadao/ejud-search*) disponibiliza todos os processos públicos, entretanto ele não apresenta a informação das pessoas envolvidas facilmente, sendo necessário desenvolver códigos para vasculhar as páginas do *site*. Em conversa com um especialista do domínio, foi fornecido um arquivo .CSV com informações sobre processos extrajudiciais e contendo três colunas sendo elas “Interessado”, “Investigado” e “Texto”.

As colunas de Interessados e Investigados apresentam os nomes das partes envolvidas no procedimento. Elas estão representadas em *string*. A coluna de texto continha *tags* HTML que dificultaram a legibilidade das informações, sendo necessário realizar um tratamento sobre eles.

Figura 10: Exemplo do *BeautifulSoup*

```
1  from bs4 import BeautifulSoup as bs
2  import csv
3
4  with open('dados_teste.CSV', newline = '', encoding = "utf-8") as csvfile:
5      arquivoCSV = csv.reader(csvfile, delimiter = ',')
6
7      for linha in (arquivoCSV):
8          texto = bs(linha[2], 'html.parser').findAll(class_ = "paragraph-text")[0]
9          texto = texto.get_text(separator = " ", strip = True)
```

Fonte: Autoria própria

A Figura 10 mostra um trecho de código utilizando duas bibliotecas *Python*, a *BeautifulSoup* e a csv. A primeira é capaz de interpretar trechos HTMLs e convertê-los em objetos de linguagem. Esses objetos conseguem retornar o texto interno do HTML, a possibilidade de encontrar determinados tipos de *tags* HTML dentro de todo o documento através de filtros e outras funções que facilitam a navegação e a obtenção de determinados pontos do HTML, já a segunda é utilizada para ler arquivos com extensão .CSV como também facilitar a manipulação das linhas e colunas presentes na tabela.

Nas linhas 4 e 5 representadas na Figura 10 acontece a leitura do arquivo “dados_teste.CSV” pelo método “open” nativo do *Python* e a geração de uma lista com os elementos das três colunas do arquivo .CSV apresentado anteriormente. Uma iteração é construída com essa lista e enfim utilizada a *BeautifulSoup* nas linhas 8 e 9 para localizar no elemento da posição número 2 todas as *tags* HTML que contenham a classe “paragraph-text” e a extração do texto interno do elemento encontrado. Essa classe é única dentro dos dados fornecidos e ela contém o texto escrito pelo autor do processo.

Para facilitar ainda a manipulação dos textos, foi utilizado um *corpus* da biblioteca NLTK que continha *stopwords* portuguesas, palavras que podem ser retiradas de textos sem prejudicar o sentido total dele. Foi usada também uma função que possuía expressões regulares para a remoção de caracteres indesejados.

Figura 11: Remoção de Caracteres Indesejados

```
1  from nltk.corpus import stopwords
2  import re, nltk
3
4  nltk.download('stopwords')
5  STOPS = set(stopwords.words('portuguese'))
6
7  def removeString(string):
8      newString = ""
9      for i in string:
10         newString += re.sub(r'[^\w\s]|[0-9]|["º_\n"]', ' ', i, flags=re.IGNORECASE)
11
12     while "  " in newString:
13         newString = newString.replace("  ", " ")
14
15     string = ''.join(i+" " for i in string.split(" ") if i not in STOPS)
16
17     return newString.strip()
18
19 texto = removeString("texto")
```

Fonte: Autoria própria

Na linha 4 da Figura 11 observa-se o *download* e o carregamento para uma variável das *stopwords* portuguesas utilizando o NLTK. Na linha 15 é realizada uma iteração entre os elementos do texto caso estes não estejam presentes na lista de *stopwords*. Entre as linhas 7 à 17 existe uma trecho de código com chamada para uma “função” que executa mais tratamentos sobre os textos, em especial nas linhas 9 e 10 onde é utilizado uma expressão regular para remoção de caracteres especiais, números, “º” e quebras de linhas. Com exceção de acentos, todos esses caracteres foram substituídos por um espaço vazio.

Com o texto tratado, foi necessário então identificar as possíveis entidades existentes dentro dele. Para isso foi usada a biblioteca gratuita *SpaCy* que trabalha com processamento de linguagem natural e que possui alguns modelos prontos para o uso.

No *website* do *SpaCy* é possível encontrar um manual de como começar a trabalhar com ela, é possível escolher entre vários idiomas, o tipo de uso e também se o conjunto de dados requeridos seja voltado à eficiência ou à acurácia.

Figura 12: Uso do *SpaCy*

```
1   import spacy
2
3   nlp = spacy.load("pt_core_news_lg")
4
5   doc = nlp("texto")
6
7   for palavra in doc.ents:
8       if palavra.label_ == 'PER' or palavra.label_ == 'ORG':
9
10          ENTITIES.append((palavra.text, count))
```

Fonte: Autoria própria

Na Figura 12 pode-se identificar na linha 3 o carregamento do *dataset* escolhido, este com nome "*pt_core_news_lg*" que traz consigo uma alta taxa de acurácia e treinado com um grande volume de dados, resultando em um dataset de aproximadamente 500MB (SPACY, 2021). A função "nlp" usada na linha 5 retorna um objeto com diversos elementos, em especial, o "ents" que possui as palavras e entidades reconhecidas. Este modelo do *SpaCy* consegue identificar diversos tipos de entidades, entretanto, para este projeto, o foco foi filtrar apenas aquelas identificadas como "PER" (pessoas) e "ORG" (organizações), e alocando uma tupla com a palavra e o número do seu processo a uma lista chamada "ENTITIES" para ser trabalhada posteriormente.

Ainda na iteração de descoberta das entidades foi utilizado um trecho de código para coletar as palavras anteriores e posteriores a ela, palavras estas que foram a base do *dataset* utilizado na rede neural.

Figura 13: Palavras Anteriores e Posteriores

```
1   splitBy     = texto.split(palavra.text)
2   wordsBefore = splitBy[0].split(" ")[-5:-1]
3   wordsAfter  = splitBy[1].split(" ")[0:5]
4
5   for word in wordsBefore:
6       if word not in wordsBefore:
7           WORDS_BEFORE.append(word)
8
9   for word in wordsAfter:
10      if word not in wordsAfter:
11          WORDS_AFTER.append(word)
```

Fonte: Autoria própria

Cada texto foi separado pela palavra da entidade reconhecida e assim coletadas as cinco palavras anteriores e as cinco palavras posteriores a ela, representado nas linhas 1, 2 e 3 da Figura 13. Essas palavras foram então atribuídas a duas listas de palavras, WORDS_BEFORE e WORDS_AFTER (palavras anteriores e posteriores, respectivamente) caso as palavras não existam dentro das listas.

De forma manual, para cada entidade reconhecida pelo *SpaCy*, foi atribuída uma categoria em formato numérico, sendo que 0, a entidade em questão não tem relação com o texto, sendo 1 ela é investigada e se 2, ela é da parte interessada. Essa categorização gerou um arquivo que foi usado pela rede neural como forma de identificação no treinamento.

4.2 Conversão dos Dados

As listas de palavras anteriores e posteriores geradas na etapa passada precisaram ser convertidas para um formato em que o computador e a rede pudessem entender. Assim novamente outra iteração entre os textos foi realizada. Desta vez, como as entidades identificadas anteriormente foram salvas, o processo foi mais rápido. Foi criada uma lista de elementos com o tamanho de todas as "*WORDS_AFTER*" mais as "*WORDS_BEFORE*" encontradas para cada entidade identificada, resultando em uma lista de 1174 elementos para cada uma das 408 entidades reconhecidas.

Figura 14: Processo de Conversão das Listas

```
1   def addIndexer(wbefore, wafter):
2       index = [0] * len(WORDS_AFTER + WORDS_BEFORE)
3
4       for num, i in enumerate(WORDS_AFTER):
5           for j in wafter:
6               if i[0] == j:
7                   index[num] = 1
8
9       for num, i in enumerate(WORDS_BEFORE):
10          for j in wbefore:
11              if i[0] == j:
12                  index[num] = 1
13
14      INDEXERS.append(index)
```

Fonte: Autoria própria

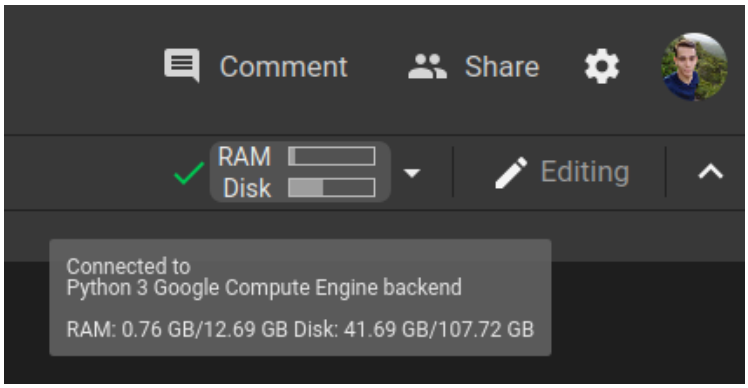
A lista para cada entidade é gerada primeiramente com todos os 1174 elementos sendo zeros. A Figura 14 demonstra o trecho em que para cada palavra anterior e posterior à entidade,

o valor zero da posição de cada palavra, se encontrado no trecho da entidade, é transformado em 1 dentro da lista.

4.3 Construção da Rede Neural

A construção da rede neural teve como forma a utilização do ambiente Google *Collaboratory*, ele é um ambiente virtualizado onde pode-se pegar "emprestado" uma máquina da Google com recursos e bibliotecas específicas para trabalhar com o desenvolvimento e treinamento de algoritmos de *machine learning* e redes neurais.

Figura 15: Recursos do Google *Colab*



Fonte: Autoria própria

A Figura 15 exibe a quantidade de memória RAM (Memória de Acesso Aleatório) e disco totais e usados que a máquina a qual estava conectado no momento oferecia, sendo possível até mudar o tipo de aceleração para conseguir foco em TPU (Unidade de Processamento de Tensores), GPU (Unidade de Processamento Gráfico) ou o balanceamento entre os dois recursos.

O modelo de construção da rede neural utilizando o *Keras* adota uma forma compactada em poucas linhas, onde é preciso definir primeiramente o tipo da rede, neste projeto foi usado o tipo *Sequential*, o qual define que a rede terá camadas em sequência, sem poder voltar camadas ou construir iterações entre elas.

Figura 16: Modelo Final da Rede Neural

```
1  import keras, tensorflow
2
3  modelo = keras.Sequential([
4    keras.layers.LSTM(units = 50, return_sequences = True, input_shape = (1, 1174)),
5    keras.layers.LSTM(units = 50, return_sequences = True),
6    keras.layers.Dense(100, activation = tensorflow.nn.relu),
7    keras.layers.LSTM(units = 50, return_sequences = False),
8    keras.layers.Dense(50, activation = tensorflow.nn.relu),
9    keras.layers.Dense(3, activation = tensorflow.nn.softmax)
10 ])
```

Fonte: Autoria própria

Dentro do módulo *Sequential*, o modelo recebe as camadas em formato de lista, o que pode ser observado da linha 4 à linha 9 da Figura 16, sendo que cada linha é uma camada da rede neural. Na primeira camada é importante definir o formato dos seus dados (*input_shape*) para assim a camadas conseguir trabalhar em cima deles, ou o processamento será interrompido por causa de erros. O formato dos dados consistiu no par de números 1 e 1174, significando uma única linha contendo 1174 colunas, sendo que cada coluna representa uma palavra da lista geral.

Foram utilizados dois tipos de camadas, as camadas LSTM que como já dito, possuem o aspecto "memória" e as camadas do tipo Densa (*dense*) que ligam todos os nós internos entre eles mesmos. Vale destacar nas camadas densas o primeiro parâmetro significa a quantidade de categorias ao qual ela terá de saída, por exemplo, esta rede neural deve dizer se a entidade não é nada no texto, se é interessado ou se é investigada, logo, são 3 categorias e este é o número que deve conter na última camada densa.

Nota-se ainda na Figura 16 a presença de duas ativações (*activations*) diferentes, a ReLU (*Rectified Linear Unit*) e a SoftMax. A primeira serve para corrigir valores negativos, transformando-os em zeros (BROWNLEE, 2019) e a segunda transforma as diferentes dimensões dos seus dados em probabilidade (SAKO, 2018).

Após criar o modelo, foi necessário compilá-lo e treinar a rede, esse processo também foi realizado dentro do *Colab*. Apesar da rede neural ter seis camadas, o treinamento não exigiu bastante do *hardware* e foi concluído em questão de segundos devido à quantidade de dados no *dataset*, visto anteriormente que constituía de 408 entidades com 1174 palavras atreladas a cada uma.

Figura 17: Treinamento da Rede Neural

```
historico = modelo.fit(master, identificacoes, epochs = 12, validation_split=0.2)

Epoch 1/12
11/11 [==============================] - 7s 143ms/step - loss: 1.0725 - accuracy: 0.8497 - val_loss: 1.0238 - val_accuracy: 0.9024
Epoch 2/12
11/11 [==============================] - 0s 11ms/step - loss: 0.9684 - accuracy: 0.8896 - val_loss: 0.8618 - val_accuracy: 0.9024
Epoch 3/12
11/11 [==============================] - 0s 11ms/step - loss: 0.7236 - accuracy: 0.8896 - val_loss: 0.4995 - val_accuracy: 0.9024
Epoch 4/12
11/11 [==============================] - 0s 11ms/step - loss: 0.4010 - accuracy: 0.8896 - val_loss: 0.3265 - val_accuracy: 0.9024
Epoch 5/12
11/11 [==============================] - 0s 11ms/step - loss: 0.3436 - accuracy: 0.8896 - val_loss: 0.2924 - val_accuracy: 0.9024
Epoch 6/12
11/11 [==============================] - 0s 13ms/step - loss: 0.2836 - accuracy: 0.8896 - val_loss: 0.2877 - val_accuracy: 0.9024
Epoch 7/12
11/11 [==============================] - 0s 12ms/step - loss: 0.2450 - accuracy: 0.8896 - val_loss: 0.2527 - val_accuracy: 0.9024
Epoch 8/12
11/11 [==============================] - 0s 13ms/step - loss: 0.2064 - accuracy: 0.8896 - val_loss: 0.2269 - val_accuracy: 0.9024
Epoch 9/12
11/11 [==============================] - 0s 13ms/step - loss: 0.1774 - accuracy: 0.8896 - val_loss: 0.2187 - val_accuracy: 0.9024
Epoch 10/12
11/11 [==============================] - 0s 12ms/step - loss: 0.1619 - accuracy: 0.8896 - val_loss: 0.2166 - val_accuracy: 0.9024
Epoch 11/12
11/11 [==============================] - 0s 12ms/step - loss: 0.1510 - accuracy: 0.8896 - val_loss: 0.2033 - val_accuracy: 0.9024
Epoch 12/12
11/11 [==============================] - 0s 12ms/step - loss: 0.1438 - accuracy: 0.9141 - val_loss: 0.1899 - val_accuracy: 0.9512
```

Fonte: Autoria própria

Observa-se que a rede treina diversas vezes, aumentando o número da geração. Este exemplo da Figura 17 foi treinado por 12 épocas e com 20% dos dados passados a ele como forma de validação se a rede está certa ou errada, de acordo com o *dataset*. Note também que é exibido outras informações como perda (*loss*) e acurácia (*accuracy*) tanto do treinamento quanto da validação. A perda representa a diferença entre o valor de saída da rede e o esperado para uma determinada entrada, ela também é usada pela rede em cada uma das gerações para ajustar os pesos e *bias* do modelo compilado (MISHRA, 2021). Observando a taxa de perda se aproximando de zero significa que a rede está obtendo cada vez mais sucesso ao fazer suas previsões.

4.4 TESTES E EXPERIMENTOS

Com a rede neural realizando uma grande quantidade de cálculos a respeito dos dados em seus nós, o consumo de recursos é inevitável. Os primeiros testes da rede com dados ainda não estruturados corretamente resultaram em comportamentos anormais para a quantidade de dados do qual estava sendo treinado.

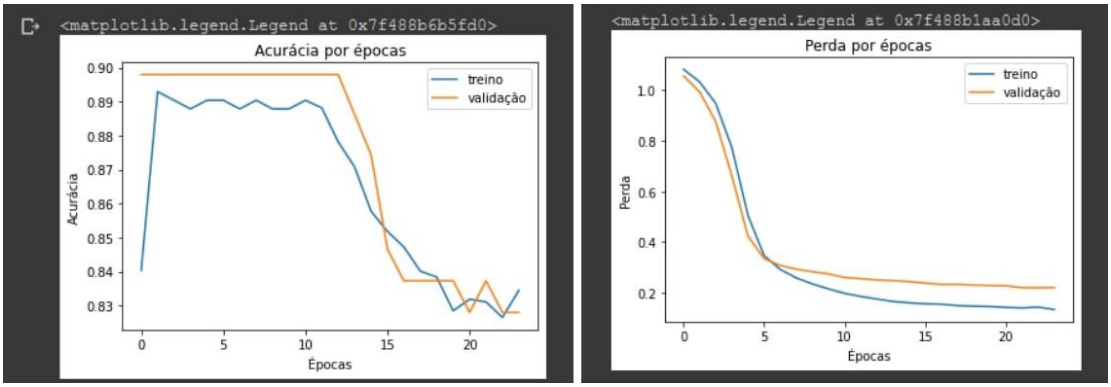
Figura 18: Recursos do *Colab* em Uso

Fonte: Autoria própria

Na Figura 18 observa-se que nem mesmo os 12GB de memória RAM disponíveis foram suficientes, com o ambiente do *Colab* sendo reiniciado inúmeras vezes por conta disso.

Mais testes foram constantemente sendo realizados a fim de melhorar tanto o desempenho, quanto a perda e a acurácia da rede. Fazendo uso da biblioteca *matplotlib* que é capaz de gerar gráficos, fica evidente a visualização da perda e da acurácia ao longo do treinamento.

Figura 19: Gráficos Decaídos de Acurácia e Perda da Rede

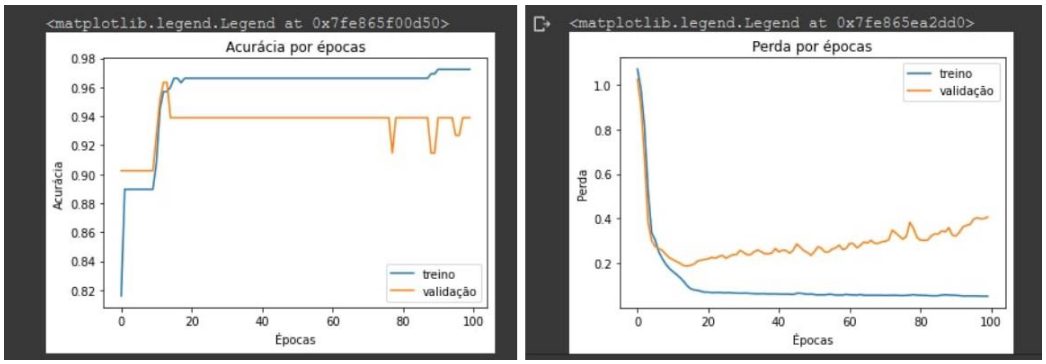


Fonte: Autoria própria

Existem casos em que treinamento demais pode piorar o desempenho da rede, como mostra a Figura 19. Pode-se notar que entre as gerações 10 e 15 houve uma queda de quase 10% na taxa de acurácia, entretanto a taxa de perda não foi afetada.

Seguindo o mesmo raciocínio sobre a quantidade de gerações para se treinar, tanto os valores de treino como de validação devem andar juntos, caso isso não ocorra, pode acontecer *overfiting* ou *underfiting* da rede. O *overfitting* acontece quando o treino vai bem, mas a validação não, a rede acaba decorando alguns padrões e tenta utilizá-los em outras variáveis, o que não funciona bem (LUCIAN, 2020). O *underfitting* acontece quando a acurácia do modelo vai ruim tanto no treinamento quanto na validação, isso porque a rede não encontra relações entre as variáveis (REHAN, 2021).

Figura 20: Gráficos Estagnados de Acurácia e Perda da Rede



Fonte: Autoria própria

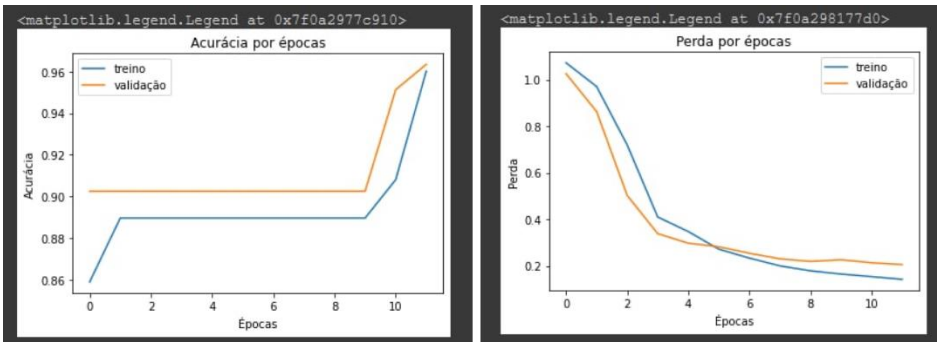
Por vezes, treinar demais pode não mudar muito a rede, como exemplo a acurácia da geração 20 à 70 da Figura 20 mostra uma estagnação, enquanto a perda na validação foi subindo constantemente.

4.5 RESULTADOS

A rede foi treinada com um *dataset* de 408 entidades, sendo que 364 foram identificadas como “Nada no texto”, 2 sendo “Investigadas” e 42 sendo “Interessadas”, ainda teve em seu modelo três camadas LSTM e três camadas densas, sendo treinadas por 12 gerações.

Após diversas tentativas de treinamento com camadas distintas e parâmetros diferentes, chegou-se a um resultado satisfatório.

Figura 21: Gráficos Finais da Rede



Fonte: Autoria própria

Como resultado dessa observação, foi percebido que dentre uma das tentativas, a parada na geração 12 foi a que se saiu melhor, tendo uma acurácia e perda de aproximadamente 96% e 20%, respectivamente, como observado na Figura 21.

A saída da rede neural é representada em uma lista contendo as três porcentagens de chance para cada categoria, por via, pode-se apresentar apenas a maior porcentagem dizendo que a entidade reconhecida é da categoria tal, ou mesmo apresentar as três para que a pessoa que usufrua da rede tome sua própria decisão.

Como uso facilitado da rede, um pequeno algoritmo em *Python* foi escrito para receber o texto do usuário, tratá-lo, identificar as entidades, coletar as palavras anteriores e posteriores a ela, converter esses dados em uma lista de zeros e uns e fazer a predição com o modelo salvo da rede neural, assim retornando um objeto que traz a predição acerca daquele texto.

Figura 22: Saída da Rede Neural


```
[
  {
    "nome": "Thiago Felipe Schuch",
    "prediçao": [
      {"interessado": 21.25},
      {"investigado": 0.09},
      {"sem relação": 51.25}
    ]
  }
]
```


Fonte: Autoria própria

A Figura 22 demonstra uma saída da rede, ela exporta um objeto JSON contendo o nome da entidade reconhecida como chave e as porcentagens de “Nada no texto”, “Investigado” e “Interessado” em uma lista como valor.

5 CONSIDERAÇÕES FINAIS

O presente trabalho descreveu o desenvolvimento de uma rede neural do tipo LSTM para identificação de interessados e investigados em procedimentos extrajudiciais.

Apesar da rede neural possuir uma boa taxa de acurácia e de perda, os problemas enfrentados durante o desenvolvimento podem ter limitado o resultado alcançado. A começar pela quantidade de dados disponíveis para o treinamento, contando com apenas 408 entidades. A rede não conseguiu entender o bastante para ter uma boa taxa de perda, visto que ela ainda ficou com aproximadamente 20%.

Outro fator decorrente ao das entidades foi a distribuição desigual das categorias, sendo que 1% das entidades do *dataset* eram “Investigadas”, 10% eram “Interessadas” e 89% não eram nem “Investigadas” nem “Interessadas”. Essa distribuição fez com que a rede aprendesse mais quem não fazia parte do texto do que apontar de fato quem era investigado e quem era interessado.

A dificuldade em tratar os dados também pode ter influenciado a rede, devido às funções de remoção de caracteres não conseguir limpar todos os textos, com certos caracteres sendo levados adiante e até mesmo palavras completas terem ficado irregulares, como a junção de palavras com hífen, transformando “noticia-se” em “noticiase” ou juntando palavras aleatórias.

Em contrapartida, a pequena quantidade de categorias possíveis pode ter ajudado a rede neural a ter um bom desempenho, fazendo com que ela alcançasse a marca de 96% de acurácia.

Outra observação é que com o grande número de itens para treino sendo classificados como interessados ou sem relação com o texto, a rede neural aprendeu e conseguiria dizer melhor quem faz parte do texto e quem não faz parte do que o objetivo proposto neste trabalho.

Como trabalhos futuros, fica sendo necessário um melhor refinamento dos textos recebidos, tanto dos usuários quanto os de novos dados para que a rede não sofra tanta interferência de ruídos. A utilização de um *dataset* maior também deverá ser levada em consideração, a fim de comparar o desempenho das duas redes utilizando o mesmo modelo criado.

REFERÊNCIAS

AKBIK, Alan *et al*. **Flair: an easy-to-use framework for state-of-the-art nlp**. Proceedings Of Naacl-Hlt. Berlim, p. 54-59. jun. 2019. Disponível em: https://aclanthology.org/N19-4010.pdf. Acesso em: 11 jul. 2021.

AKBIK, Alan *et al.* **Contextual string embeddings for sequence labeling**. Proceedings of the 27th International Conference on Computational Linguistics, p. 1638–1649. ago. 2018. Disponível em: https://alanakbik.github.io/papers/coling2018.pdf. Acesso em: 11 jul. 2021.

ALVAREZ, A.; LUQUE, B, 2003, Campinas. **Rede neural de kohonen e outras técnicas para treinamento não-supervisionado.** Universidade Estadual de Campinas, p. 34, 2003. Disponível em: ftp://calhau.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_03/revisao/docs/tema4.doc. Acesso em: 15 abr. 2021.

ARAÚJO, Pedro Henrique Luz de *et al*. **LeNER-Br: a dataset for named entity recognition in brazilian legal**. Computational Processing Of The Portuguese Language, Brasília, p. 313-323, set. 2018. Disponível em: https://cic.unb.br/~teodecampos/LeNER-Br/luz_etal_propor2018.pdf. Acesso em: 13 jul. 2021.

BATISTA, Gustavo Enrique de Almeida Prado Alves. **Pré-processamento de Dados em Aprendizado de Máquina Supervisionado**. 2003. 232 f. Tese (Doutorado) - Curso de Ciências de Computação e Matemática Computacional, Icmc-Usp, São Carlos, 2003. Disponível em: https://www.teses.usp.br/teses/disponiveis/55/55134/tde-06102003-160219/publico/TeseDoutorado.pdf. Acesso em: 12 jun. 2021.

BRASIL. Constituição (2006). **Lei Nº 11.419**. Brasília, 19 dez. 2006. Disponível em: http://www.planalto.gov.br/ccivil_03/_Ato2004-2006/2006/Lei/L11419.htm. Acesso em: 24 abr. 2021.

BROWNLEE, Jason. **A Gentle Introduction to the Rectified Linear Unit (ReLU)**. 2019. Disponível em: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. Acesso em: 4 dez. 2021.

CHAUDHARY, Makesh. **Activation Functions**: sigmoid, tanh, relu, leaky relu, softmax. Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax. 2020. Disponível em: https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5. Acesso em: 24 abr. 2021.

CONGRESSO BRASILEIRO DE ENERGIA SOLAR, 8., 2020, Fortaleza. **Previsão de Geração Fotovoltaica a Partir de Dados Meteorológicos Utilizando Rede LSTM.** Fortaleza: Universidade Federal de Alagoas, 2020. Disponível em: https://anaiscbens.emnuvens.com.br/cbens/article/view/761/761. Acesso em: 24 abr. 2021.

CONSELHO NACIONAL DE JUSTIÇA (CNJ) (Brasília). **Justiça em Números 2020**: nova edição confirma maior produtividade do judiciário. Nova edição confirma maior produtividade do Judiciário. 2020. Disponível em: https://www.cnj.jus.br/justica-em-numeros-2020-nova-edicao-confirma-maior-produtividade-do-judiciario. Acesso em: 13 abr. 2021.

CHOWDHURY, Gobinda, 2003, Glascow. **Natural Language Processing.** Glascow: University Of Strathclyde, 2003. 39 p. Disponível em: https://strathprints.strath.ac.uk/2611/1/strathprints002611.pdf. Acesso em: 5 jun. 2021.

DUARTE, João, 2017, Lisboa. **Sebenta de Teoria da Norma Jurídica.** Lisboa: Universidade Nova de Lisboa, 2017. Disponível em: https://ae.fd.unl.pt/wp-content/uploads/2019/10/Sebenta-teoria-Norma-Juridica-Joao-Duarte.pdf. Acesso em: 13 abr. 2021.

EBERHARD *et al*. **Ethnologue**: languages of the world. Languages of the World. 2021. Disponível em: https://www.ethnologue.com. Acesso em: 20 jun. 2021.

HENRIQUES, Claudio Cezar. **Léxico e Semântico**: estudo produtivos sobre palavra e significação. Rio de Janeiro: Alta Books, 2018. Disponível em: https://books.google.com.br/books?id=X_xmDwAAQBAJ. Acesso em: 21 jun. 2021.

KOVÁCS, Zsolt László. **Redes Neurais Artificiais**: fundamentos e aplicações. 4. ed. São Paulo: Livraria da Física, 2006. Disponível em: https://books.google.com.br/books?id=O0nLxR67wmUC. Acesso em: 5 jun. 2021.

KUMAR, Ela. **Natural Language Processing**. Nova Delhi: I. K. International Publishing House Pvt. Ltd., 2010. Disponível em: https://books.google.com.br/books?id=FpUBFNFuKWgC. Acesso em: 21 jun. 2021.

LUCIAN, Bruno. **Overfitting: o que é e como evitar**. 2020. Disponível em: https://www.dadosaleatorios.com.br/post/overfitting/. Acesso em: 5 dez. 2021.

MARRERO, Mónica *et al*. Named Entity Recognition: fallacies, challenges and opportunities. **Computer Standards & Interfaces**, [S.L.], v. 35, n. 5, p. 482-489, set. 2013. Elsevier BV. http://dx.doi.org/10.1016/j.csi.2012.09.004. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0920548912001080. Acesso em: 5 jun. 2021.

MATSUNAGA, Victoria Yukie. **Curso de Redes Neurais utilizando o MATLAB**. Belém, 2021. Disponível em: https://kupdf.net/download/redes-neurais-apostilapdf_5c521770e2b6f5e31caba802_pdf. Acesso em: 24 abr. 2021.

MIKHEEV, Andrei *et al*. **Named Entity Recognition without Gazetteers.** Groningen: University Of Groningen, Groningen. 1999. Disponível em: https://aclanthology.org/E99-1001.pdf. Acesso em: 11 jul. 2021

MISHRA, Ayush. **Overfitting, Losses, and Accuracies of a Neural Network Model: A General Visualisation**. 2021. Disponível em: https://ayushvns.medium.com/overfitting-losses-and-accuracies-of-a-neural-network-model-a-general-visualisation-2e3e90449d8. Acesso em: 4 dez. 2021.

OLAH, Christopher. **Understanding LSTM Networks**. 2015. Disponível em: http://colah.github.io/posts/2015-08-Understanding-LSTMs. Acesso em: 24 abr. 2021.

PEIXOTO, Sirlene Maria. **Impactos Do Processo Eletrônico E-Proc no MPF – PRR4ª Região**. 2016. 81 f. Dissertação (Mestrado) - Curso de Administração Pública Contemporânea, Ciências Administrativas, Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2016. Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/156482/001016196.pdf. Acesso em: 24 abr. 2021.

PEREIRA, Maria Neuma. **Processo Digital**: a tecnologia aplicada como garantia da celeridade processual. São Paulo: Biblioteca24Horas, 2011. 221 p. Disponível em: https://books.google.com.br/books?id=kpyTkPnDxlIC. Acesso em: 24 abr. 2021

PINTO, Danilo César Souza. **Em Nome da Segurança e da Desconfiança**: um estudo antropológico sobre o funcionamento da burocracia. 2007. 118 f. Dissertação (Mestrado) - Curso de Ciências Sociais, Universidade Federal de São Carlos, São Carlos, 2007. Disponível em: https://repositorio.ufscar.br/bitstream/handle/ufscar/1450/DissDCSP.pdf. Acesso em: 25 abr. 2021.

RABELO, Styphany Ferreira. **O Processo, os Meios Eletrônicos na Esfera Processual e o E-Proc**. 2009. 56 f. TCC (Graduação) - Curso de Ciências Jurídicas, Universidade Federal de Santa Catarina, Florianópolis, 2009. Disponível em: https://repositorio.animaeducacao.com.br/bitstream/ANIMA/7473/1/98216_Styphany.pdf. Acesso em: 24 abr. 2021.

REHAN. **Overfitting and Underfitting in Deep Learning**. 2021. Disponível em: https://ainewgeneration.com/overfitting-and-underfitting-in-deep-learning/. Acesso em: 5 dez. 2021.

ROSENBLATT, Frank. **Perceptron Simulation Experiments. Proceedings of the IRE.** 1960. 48(3), 301–309. doi:10.1109/jrproc.1960.287598. Disponível em: https://sci-hub.st/10.1109/JRPROC.1960.287598. Acesso em: 24 abr. 2021.

SAKO, Yusako. **Is the term "softmax" driving you nuts?** 2019. Disponível em: https://medium.com/@u39kun/is-the-term-softmax-driving-you-nuts-ee232ab4f6bd. Acesso em: 4 dez. 2021.

SARDINHA, Tony Berber. **Linguística de Corpus**. Barueri: Manole, 2004. Disponível em: https://books.google.com.br/books?id=i8uJXgeok48C. Acesso em: 29 jun. 2021.

SILVA, Douglas da. **Processamento de linguagem natural: entenda como funciona, importância e aplicação [Guia Completo]**. 2021. Disponível em: https://www.zendesk.com.br/blog/processamento-de-linguagem-natural/. Acesso em: 5 dez. 2021.

SIMÕES, Alexandre da Silva. **Aprendizado não-supervisionado em redes neurais pulsadas de base radial**. 2006. 184 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Escola Politécnica da Usp, São Paulo, 2006. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3141/tde-15092006-153353/publico/AlexandreDaSilvaSimoes.pdf. Acesso em: 12 jun. 2021.

SPACY. **Spacy**. Disponível em: spacy.io. Acesso em: 13 jul. 2021.

TEIXEIRA, Wagner Rodrigues *et al*. **Exemplo de Extração de Definições em Textos Articulados de Normas Jurídicas com o Apoio do Processamento de Linguagem Natural**. Cajur, Brasília, v. 1, n. 6, p. 49-64, jun. 2019. Disponível em: http://www.cajur.com.br/index.php/cajur/article/download/212/288. Acesso em: 13 jul. 2021.

WANG, Bin *et al*. **Evaluating word embedding models**: Methods and experimental results. 2019. APSIPA Transactions on Signal and Information Processing, 8, E19. doi:10.1017/ATSIP.2019.12. Disponível em: https://www.cambridge.org/core/journals/apsipa-transactions-on-signal-and-information-processing/article/evaluating-word-embedding-models-methods-and-experimental-results/EDF43F837150B94E71DBB36B28B85E79. Acesso em: 29 jun. 2021.