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.

Jonhtan Mota Dos Reis

REDE NEURAL PARA IDENTIFICAR ÀREA DE ATUAÇÃO DE PROCEDIMENTOS EXTRAJUDICIAIS

Palmas – TO

2021

Jonhtan Mota Dos Reis

REDE NEURAL PARA IDENTIFICAR ÀREA DE ATUAÇÃO DE PROCEDIMENTOS EXTRAJUDICIAIS

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo.

Palmas – TO

2021

Jonhtan Mota Dos Reis

REDE NEURAL PARA IDENTIFICAR ÀREA DE ATUAÇÃO DE PROCEDIMENTOS EXTRAJUDICIAIS

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Esp. Fábio Castro Araújo

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Madianita Bogo Marioti

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2021

AGRADECIMENTOS

Agradeço aos meus familiares pelo apoio e incentivo que me deram dês do início da minha jornada, por mais que tenha ficado difícil ou complicado, poder olhar para cada um deles e saber que teria em quem me apoiar foi o que sempre me fez seguir em frente sem medo.

Agradeço a minha mãe, Alessandra, por sempre estar do meu lado, por não me deixar desistir, por me incentivar a ser quem eu sou, por mais que não conseguia entender muito bem com o que eu trabalhava, muito obrigado por me apoiar.

Agradeço meu irmão, Maycon, por sempre estar do meu lado, crescemos juntos e compartilhamos essa paixão por computadores dês de cedo, sempre falamos como seria incrível fazer um computador executar ações e nossos comandos, criar aplicativos e jogos. E em busca de tornar isso realidade pude escolher esse caminho, muito obrigado. Agradeço a minha irmã, Sara, obrigado por me demonstrar um outro lado mais empático para com os outros, a entender melhor as coisas ao meu redor.

Agradeço aos meus colegas e amigos de faculdade que me acompanharam dês do primeiro dia em que pisei na universidade, não sabia o que me esperava e estava bem nervoso, se eu soubesse que encontraria pessoas tão incríveis não teria me preocupado tanto.

Agradeço a todos os professores que compartilharam seu conhecimento e sabedoria comigo, em especial meu orientador, Fábio Castro, pelas suas orientações e amizade, grande responsável por trazer as alegrias nas noites a fio de aulas intensas. E agradeço a todos aqueles que estiveram comigo nesse decorrer de 4 anos de curso.

RESUMO

REIS, Jonhtan Mota dos. **Rede Neural para Identificar Área De Atuação De Procedimentos Extrajudiciais**. 2021. 50 f. Trabalho de Conclusão de Curso (Graduação) - Curso de Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2021.

O presente trabalho tem por proposta a implementação de uma rede neural capaz de identificar áreas de atuação de documentos extrajudiciais, utilizando-se de um modelo de aprendizado recorrente conhecido como LSTM. Onde a rede neural é capaz de aprender a partir de entradas passadas, assim possuindo a característica de memória, aprendendo com seus erros. No processo foi necessário compreender conceitos como modelagem de *dataset*, que consiste em técnicas para analisar e compreender textos em busca de categorias que possam ser utilizadas em um contexto de identificação. Fazendo-se o uso de algoritmos para identificação de palavras recorrentes dentro dos textos extrajudiciais, para reconhecer características semelhantes dentro dos textos dos processos extrajudiciais. Essa abordagem foi utilizada para compor um ranqueamento das palavras de maior frequência dentro dos documentos de determinadas áreas de atuação. Esse ranqueamento foi utilizado como parâmetro para construção do *dataset* de palavras com maior grau de ocorrência, criando uma quantificação de aparição das palavras ranqueadas dentro dos documentos extrajudiciais com a finalidade de ser utilizado no treinamento e implementação da rede neural com o auxílio da API do TensorFlow, o Keras. A rede neural desenvolvida neste trabalho cumpriu com sua proposta de identificar a área de atuação dos documentos extrajudiciais.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ERP	Enterprise Resource Planning
KNN	K Nearest Neighbor
LSTM	Long Short-term Memory
NLG	Natural Language Generation
NLU	Natural Language Understanding
RNN	Recurrent Neural Network
SNARC	Stochastic Neural Analog Reinforcement Calculator

SUMÁRIO

1. INTRODUÇÃO

Um dos primeiros artigos de pesquisa apresentados com relação a entidades nomeadas data de 1991 por Lisa F. Rau (1991), na *Proceedings The Seventh IEEE Conference on Artificial Intelligence*. Neste artigo Lisa expõe a necessidade de identificar palavras desconhecidas em textos com linguagem natural, onde a partir dessa necessidade foi desenvolvido um algoritmo capaz de realizar o reconhecimento de entidades nomeadas dentro de textos de jornais de notícias com relação ao mercado financeiro. O objetivo do artigo era identificar palavras desconhecidas cujas quais representavam entidades de nomes de empresas, onde foram obtidos resultados consistentes indicando que este problema poderia ser solucionado com o uso de reconhecimento de entidades nomeadas (RAU, 1991).

O Reconhecimento de Entidades Nomeadas (NER) consiste em um processo para solucionar o problema de reconhecer referências a entidades que podem variar de nomes pessoais, organizações e até países (MOHIT, 2014). Aliado ao NER, tem-se o processo de extração de informação (IE), que se trata de um procedimento de busca e extração de dados em meio a informações não estruturadas. As informações extraídas podem ser utilizadas para alimentar uma rede neural com o objetivo de identificar padrões e tentar prever resultados.

Em busca de processar informações de uma forma semelhante à da mente humana, tem-se as redes neurais, que consistem, de maneira geral, em máquinas projetadas para modelar a forma como um cérebro humano executa uma tarefa ou uma função de interesse, utilizando programação computacional (HAYKIN, 2007).

O Ministério Público do Estado do Tocantins, dentro do seu sistema de ERP (Sistema integrado de gestão empresarial) *Athenas*, desde 2016 conta com um módulo chamado *e-Ext* que é utilizado por seus órgãos internos (PROCURADORIA-GERAL DE JUSTIÇA, 2016). O módulo tem como objetivo substituir a autuação, o registro e a tramitação física de documentos extrajudiciais, além de realizar a coleta de dados estatísticos para fins de planejamento.

O *e-Ext* pode ser acessado pelo site do Ministério Público do Estado do Tocantins, onde é possível visualizar os documentos de procedimentos extrajudiciais desde que eles não sejam sigilosos. A partir da visualização de um documento de procedimento extrajudicial por meio do sistema *e-Ext*, é possível identificar sua área de atuação do documento, sendo o campo que determinará qual promotoria dará continuidade ao processo daquele documento, sendo assim a promotoria especializada naquela área de atuação lida com aquele assunto em específico determinado pela área de atuação do documento. As áreas de atuação são definidas pela corregedoria e estão divididas em categorias de macro áreas sendo elas: Cidadania,

Consumidor, Controle externo da atividade policial, Crimes contra a administração pública, Crimes contra a ordem econômica e tributária, Criminal geral, Educação, Eleitoral, Execução penal, Família, Habitação, Idosos, Infância e juventude, Meio ambiente, Patrimônio público, Pessoas com deficiência, Registro público, Saúde pública e urbanismo. Em cada área de atuação o ministério público tem como dever zelar pelos direitos do indivíduo indispensáveis, como a saúde, a vida e a dignidade.

A área de atuação do Ministério Público ligado à saúde como exemplo, onde o promotor de justiça fiscaliza a qualidade, eficiência e o acesso aos serviços ligados à área da saúde. O Ministério Público atua como o fiscal da lei para defender o direito do cidadão de ter acesso aos serviços ligados à área da saúde independente de sua condição social ou financeira.

O Ministério Público do Estado do Tocantins, utiliza procedimentos extrajudiciais e documentos de instauração eletrônicos. Nos últimos 4 anos houve uma crescente quantidade de novos procedimentos instaurados, onde em 2017 foram criados 3.875 processos, essa quantidade de processos cresceu exponencialmente, chegando em 8.169 processos criados em 2020.

A tarefa de classificação da área de atuação de procedimentos extrajudiciais no Ministério Público do Estado do Tocantins por meio do sistema *e-Ext* é feita de maneira que é necessário um cartório de registro para analisar os documentos de fato, identificando o tipo de procedimento e a localidade, então o sistema realiza um sorteio onde é realizada uma identificação da classificação do tipo de procedimento para escolher para qual promotoria é encaminhado o procedimento. Tendo em vista que os documentos necessitam de atenção especial por se tratar de documentos de meio jurídico, esse trabalho é feito de maneira manual, e demanda muito tempo para classificar os procedimentos com base em seus documentos.

Diante deste cenário, torna-se interessante o desenvolvimento de uma rede neural que possibilite remover um passo a mais que seria necessário ser realizado de maneira manual, agilizando e otimizando a forma como os procedimentos são identificados. Visto que uma rede neural pode ser treinada para compreender contexto, diferente de um processo codificado baseado em regras, onde a rede pode ser especializada e moldada para compreender o meio em que a mesma é inserida. Ainda sabendo que os documentos de trâmite são específicos do meio jurídico e necessitam de atenção especial dada sua natureza.

Foi então proposto o problema de pesquisa para este trabalho de como automatizar a identificação de áreas de atuação judiciais dentro de documentos de instauração dos procedimentos extrajudiciais. Esse problema de pesquisa gerou uma hipótese de que a utilização de uma rede neural e com o auxílio de um *dataset* com dados extraídos dos textos

de linguagem jurídica, é possível identificar e classificar a área de atuação do procedimento extrajudicial.

Então este trabalho teve como objetivo criar uma rede neural com a capacidade de identificar e classificar a área de atuação dentro de documentos de procedimentos extrajudiciais. Através do modelamento de uma rede neural com a capacidade de identificar a área de atuação de um documento extrajudicial, foi implementado uma rede neural a partir da criação de um *dataset* para servir para o treinamento da rede neural para identificar a área de atuação do documento extrajudicial.

2. REFERENCIAL TEÓRICO

2.1. REDES NEURAIS

As redes neurais são máquinas projetadas para criar modelos que buscam uma maneira de processar informações inspiradas na maneira como o cérebro humano processa informações de forma paralela. Um modelo básico de rede neural contém uma camada de neurônios, também conhecidos como "*nodes",* que servirão como entrada de dados; uma camada de neurônios chamada de "*hidden*", com o papel de atribuir um peso aos neurônios e direcioná-los para funções específicas; e, por último, tem-se a camada de saída de neurônios, onde é apresentado o resultado dos "*nodes*" processados (WANG, 2003).

Um dos primeiros trabalhos referentes a neurocomputação datam de 1943, com o trabalho do neuroanatomista e psiquiatra Warren McCulloch, *A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*. O artigo apresenta uma proposta sobre relações entre a lógica proposicional e os eventos de atividades neurais, onde tem-se a problemática no desenvolvimento de uma estrutura de rede que se assemelhasse ao procedimento de aprendizado dos neurônios através de ligações de sinapses (McCulloch, 1943).

A partir de 1951 as redes neurais entraram em uma espécie de era de ouro, com a criação do primeiro neurocomputador chamado *SNARC*, por Marvin Minsky, que possuía a habilidade de calcular e ajustar de forma automática os pesos, era um sistema em maquinário de aprendizagem que possuía características randômicas, que inspirou ideias de estruturas futuras (O'REGAN, 2013).

Em 1974 um estudante de *Harvard* chamado Paul Werbos, desenvolveu para sua dissertação um método chamado "*Backpropagation of error",* ou como ele também se refere *"Dynamic Feedback"* (WERBOS, 1994), que passou despercebido por uma década, e nos dias atuais passou a ser considerados um dos algoritmos mais importantes da história.

O método *Backpropagation* consiste em três etapas de procedimentos para adaptar uma rede neural sendo elas: um sistema de avaliação de saída, que irá avaliar o quão com sucesso a rede neural maximiza ou minimiza alguma coisa; um mecanismo de feedback dinâmico que irá calcular os derivados dos erros ou perdas em relação às saídas intermediárias; e pesos dentro da rede e um método de junção que irá responder essas derivadas adaptando parâmetros (WERBOS, 1988).

No ano de 1983 surgiu um modelo de rede neural que reconhecia numerais árabes escritos à mão, o modelo foi chamado de *"Neocognitron"*, elaborado por Fukushima, Miyake e Ito. A problemática a ser sanada era criar um modelo de rede neural com capacidade de reconhecimento de padrões visuais semelhantes ao da mente humana, para poder estudar sobre o mecanismo de reconhecimento visual do cérebro (FUKUSHIMA, ITO, & MIYAKE, 1983). Neste modelo, eles encontraram duas células provenientes do córtex visual primário, essas células chamadas de células simples e complexas, lidam com fatores visuais de processamento. O modelo "*neocognitron*" serviu de inspiração para o que temos hoje com as "*Convolutional Neural Network*" (CNN) que são as redes voltadas para a análise de imagens.

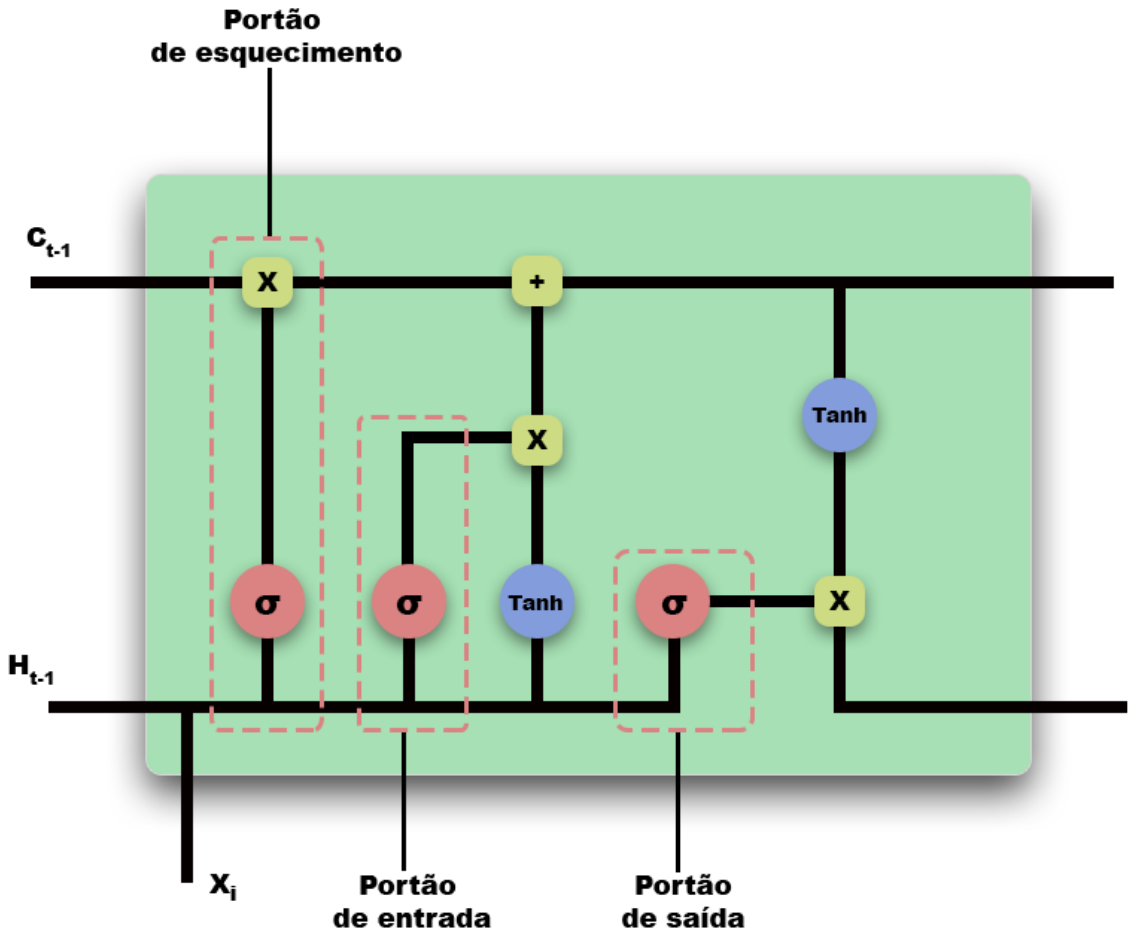
Em 1990 houve um grande foco de pesquisa na área de redes neurais direcionado a um tipo de rede neural chamada "*Recurrent Neural Network*" (RNN), modelada para aprender a partir de padrões sequenciais ou variantes. A rede conta com um sistema de conexão com *loop* de *feedback*, onde os neurônios têm acesso a informações das camadas anteriores e operações que eles mesmos realizaram anteriormente através de unidades de soma que ficam na camada *hidden* (MEDSKER, 2001).

As redes neurais do tipo RNN possuíam um problema com relação ao treinamento, por mais que elas podiam aprender baseando-se em operações anteriores, os fatores de variação, conhecidos como gradientes, desapareciam com o tempo, o que descartaria a maior vantagem desse tipo de rede, o fator de contar com informações provenientes de camadas anteriores. Com base nisso foi implementada uma rede que é considerada uma evolução do RNN, a "*Long Short-Term Memory*" (LSTM) (SHERSTINSKY, 2020).

A rede do tipo LSTM possui basicamente a mesma estrutura da rede do tipo RNN, porém, com uma substituição dentro da camada *hidden*, que ao invés de possuir unidades de soma na camada, ela passa a possuir blocos de memória. Os blocos de memória podem ser considerados semelhantes a chips de memória de um computador, onde é possível armazenar e acessar informação por longos períodos de tempo, além de remover essa informação. Dentro do bloco de memória do LSTM, é controlado de forma exclusiva o fluxo de memória através de portões de entrada, saída e esquecimento.

A figura a seguir apresenta a estrutura dentro da camada de uma rede do tipo LSTM, representado como o fluxo de navegação dos vetores de informação passam e como são processados dentro dos portões de controle de memória.

Figura 1: Estrutura da rede neural LSTM.



Como apresentado na figura 1, é possível identificar a "*cell state*", que é o controle sobre o vetor de genes que estão trafegando pela rede neural, sendo ela um estado anterior da célula que se torna um candidato para saída da rede. Logo na parte inferior do modelo é realizado uma junção de informações, onde são recebidos o atual *input* e o último "*hidden state*", sendo este os valores importantes que a rede classificou como necessários em *inputs* anteriores.

Ainda na figura 1 é possível identificar dois tipos de função de ativação, a *Sigmoid* e a *Tanh*, as funções de ativação nos portões tem a função de regular os valores da *cell state*. A função de ativação sigmoid tem como função regular os valores do vetor entre 0 e 1, o que é útil no controle de informação, dado que multiplicar um número por 0 é 0, o que faz com que os valores desapareçam do vetor, logo os valores são esquecidos. Semelhante a função de ativação sigmoid, a tanh regula os valores do vetor que fluem pela rede, porém ela garante que os valores se mantenham entre -1 e 1, pois conforme os vetores de informação navegam

pela rede sempre tendo suas informações transformadas pelas operações realizadas, os valores tendem a sair do controle, o tanh mantém esse controle.

O controle dos portões dentro do bloco de memória é o que permite o armazenamento de informação por um maior período de tempo, já que para conservar uma informação por um longo período de tempo para ser acessado eventualmente, é apenas necessário fechar o portão de entrada, assim a informação não será sobrescrevida no decorrer do tempo, diferente do caso de uma rede do tipo RNN, que ao decorrer do tempo a camada *hidden* será eventualmente sobrescrevida (GRAVES, 2012).

2.2. Processamento de Linguagem Natural

O Processamento de Linguagem Natural (NLP) consiste no estudo matemático e computacional de diversos aspectos de análise de linguagem e desenvolvimento abrangente de sistemas (JOSHI, 1991). O NLP é um subcampo da computação que estuda a capacidade de um computador aprender, criar e entender conteúdo de linguagem humana, usado principalmente de duas formas, "*Machine Translation*" (MT), que visa o uso de software para traduzir textos de uma linguagem para outra (BUDIANSKY, 1998), e beneficiar seres humanos e máquinas através de análises e aprendizado utilizando-se de grandes quantidades de conteúdos de linguagem disponíveis hoje online, conhecidos como "*Big data*" (HIRSCHBERG, 2015).

Existem inúmeros modelos de compreensão de linguagem, como os modelos "Natural Language Understanding" (NLU), que se trata de um modelo que é treinado através de um *dataset* para mapear significado, dado um *input* de texto ele utiliza-se de mecanismos denominados: intenção e entidades (JURASKY, 2000). A intenção é o que reconhece o que levou aquele determinado *input, visando* compreender o que levou àquela frase, e as entidades são objetos de uma intenção ou complemento de uma intenção.

Outro modelo é o "Natural Language Generation" (NLG), que ao contrário do NLU, tem como objetivo mapear o significado de algo para texto, logo NLG gera texto em linguagem natural que possa ser compreendido (REITER, & DALE, 2000). O NLG é amplamente utilizado em áreas de marketing e atendimento geral automatizado para o público, onde é interpretado texto em linguagem natural e a partir desse texto gerado uma resposta com o uso de NLG.

2.3. Meio Digital de Documentos Judiciais

Com a tendência mundial de digitalização ligada a globalização, ficou-se visível a necessidade de adaptação do Poder Judiciário de forma que processos de caráter jurídico adaptassem suas atividades de produção, trâmite, armazenamento e disponibilização para o meio digital, cumprindo-se seu princípio de levar acesso à justiça a todos, como diz o art. 5° da constituição brasileira de 1988 (BRASIL, 1988).

O grande marco de transição digital do Poder Judiciário data de 2006, com a Lei 11.419 de 2016, dispõe no art. 1°que será admitido o uso de processo eletrônico na tramitação de processos judiciais, comunicação de atos e transmissão de peças processuais (BRASIL, 2006), com isso teve-se o reconhecimento como meio digital, armazenamento ou tráfego de documento e arquivos digitais.

Com o avanço do meio digital e documentos de caráter jurídico passando a serem digitalizados e disponibilizados em servidores internos distribuídos nas entidades governamentais, foram apontadas melhorias no trâmite dos documentos, os procedimentos passaram a contar com disponibilidade e armazenamento facilitado. Onde por estarem sendo movidos para o meio digital, tal disponibilidade por integração de ferramentas digitais em formatos universais para tratamento dos documentos como arquivos disponíveis em formato *pdf*, aumentaram a eficiência e agilidade dentro do rito processual, por se tratar de uma estrutura consistente e simplificada.

3. METODOLOGIA

Esta seção descreve o desenho do estudo a ser realizado através de pesquisa em relação a documentos de procedimentos extrajudiciais e desenvolvimento de uma rede neural capaz de identificar a área de atuação do documento.

3.1. MATERIAIS

O primeiro passo foi utilizar *scripts* (Códigos de automação de tarefas) na linguagem de programação Python, onde o mesmo processa textos planos de descrição de atos que foram extraídos dos documentos extrajudiciais adquiridos através do sistema *E-ex*t do Ministério público, em uma busca por padrões dentro dos documentos.

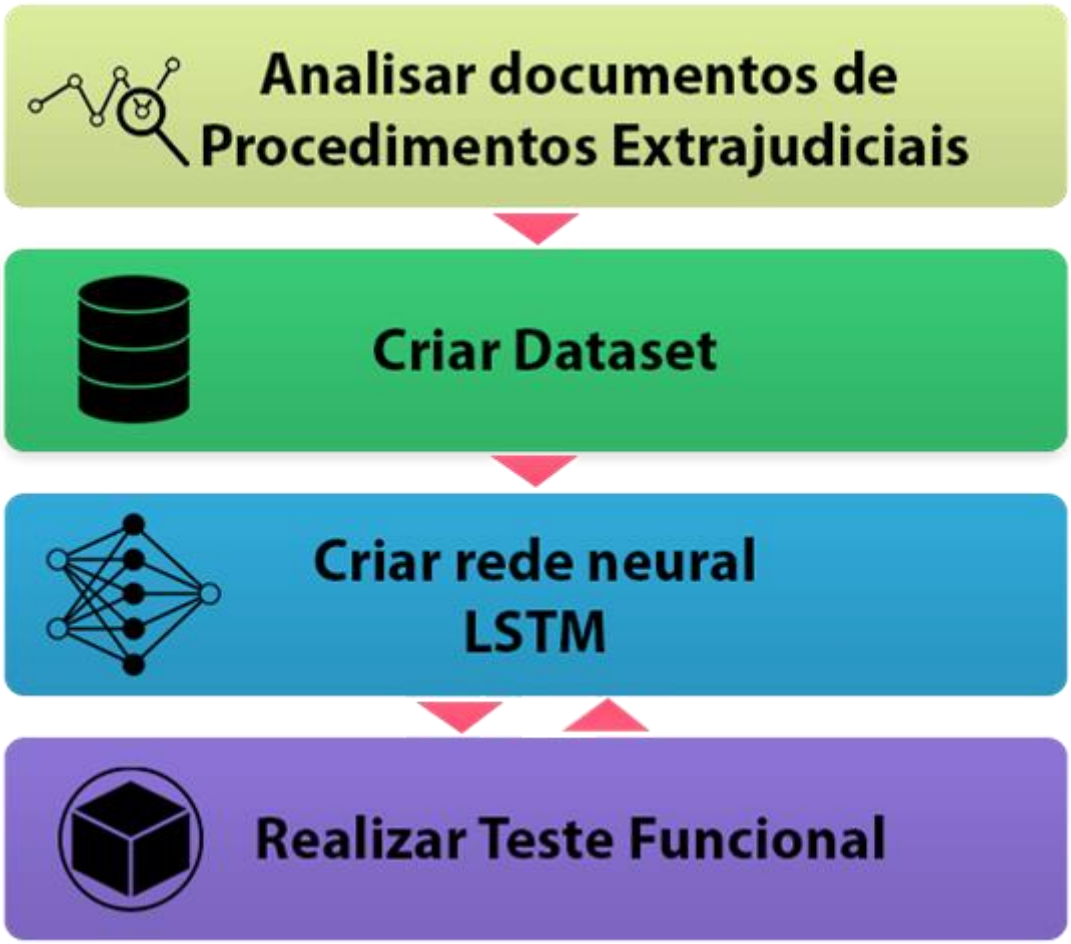
Foi utilizado um ambiente configurado sobre o Jupyter Notebook, que é uma espécie de caderno de anotação para códigos, para poder rodar as bibliotecas sem o risco de interferência externa. O ambiente foi configurado usando uma biblioteca do python chamada Anaconda, que serve para criar ambientes encapsulados com pacotes configurados pelo usuário. Este ambiente contou com os pacotes do python TensorFlow e Keras API, assim como o Pandas para poder ajustar os dados de entrada e saída de acordo com as necessidades da rede.

Na implementação da rede foi utilizado a linguagem de programação Python, compondo toda a estrutura base do projeto codificado, onde foi feito a intercomunicação com a API do Keras usando o TensorFlow. O Tensorflow é uma plataforma aberta extensiva para o desenvolvimento de soluções ligadas a *machine learning*, para atender as necessidades do projeto de rede para realizar as identificações de área de atuação dos documentos, foi implementado a rede neural do tipo LSTM (*Long Short Term Memory*).

3.2. MÉTODOS

A figura 2 apresenta os passos seguidos para o desenvolvimento da solução da rede neural deste projeto, seguindo as etapas desde a análise dos documentos extrajudiciais à concepção da rede neural.

Figura 2: Metodologia a ser utilizada.



O fluxograma abordou a forma como a pesquisa e desenvolvimento de uma rede neural que busca solucionar um problema segue. Para isso as etapas foram sequências, com ressalva para a última etapa de teste funcionais, onde foi realizado uma troca de feedbacks contínuos entre a criação da rede neural, buscando o aprimoramento da rede.

3.2.1. Analisar Documentos de Procedimentos Extrajudiciais

Através de documentos de procedimentos extrajudiciais públicos, obtidos através do site do Ministério Público do estado do Tocantins, foi realizado uma pesquisa sobre os tipos dos documentos e suas variações sendo suas áreas de atuação e protocolizações, onde foi traçado semelhanças e diferenças em busca de atribuir características únicas que podiam demonstrar padrões e estruturas.

Para identificar as estruturas e semelhanças nos textos com a finalidade de encontrar padrões ou ocorrências frequentes nos textos, foi criado um algoritmo utilizando-se de heurística de sistema de ocorrência de palavras. Essa heurística consiste em categorizar as

entradas com base em palavras semelhantes sem a necessidade de treinamento. Os dados foram processados e categorizados pelo algoritmo e então as ocorrências de maior frequência daquela determinada categorização foram salvas para servirem como parâmetro de treinamento para a rede neural.

Foram documentadas as características semelhantes dos documentos extrajudiciais visando auxiliar no processo de identificação da área de atuação dos documentos, para futuramente contribuir com o procedimento de testes de validação sobre os resultados obtidos pela rede neural. Dessa forma, ampliando o processo de extração de informações úteis que posteriormente foram adicionadas ao *dataset* para complementar os dados.

3.2.2. Criar um dataset

Nesta etapa foi elaborado e criado um *dataset* composto pelas características semelhantes como a quantidade de aparição de determinadas palavras, adquiridas nas etapas anteriores, as quais foram extraídas de textos de documentos extrajudiciais. Estas características são os tipos de áreas recorrentes dentro dos documentos, além de textos auxiliares que comunguem com a categorização da área do dito documento.

Através de um script desenvolvido para fazer a mesclagem dessas características extraídas dos documentos extrajudiciais, para poder ser analisada dentro de um documento *csv*. O arquivo possuía as linhas processadas dos arquivos apontando as ocorrências de determinadas palavras que apareciam de forma recorrente. O que foi fundamental para a análise do documento para posteriormente ser utilizado para compor o *dataset* final que seria usado como treino da rede neural.

Analisando o arquivo gerado pelo script em busca de ocorrências de palavras em comum entre as diferentes áreas de atuação, o arquivo *csv* foi convertido para uma estrutura de dados que podia ser utilizada pela rede neural. A estrutura que o arquivo passou a assumir foi a de um vetor que continha colunas fixas, possuindo valores quantificados distribuídos, representando a ocorrência ou a não ocorrência daquele determinado parâmetro.

O *dataset* final foi armazenado em formato *csv*, por se tratar de um modelo de dados bem estruturado e dispor de diversas ferramentas para a manipulação e adequação do mesmo. Este formato de arquivo também apresentou uma boa integração com as ferramentas que a linguagem *python* que foi utilizado na rede neural foi implementada.

3.2.3. Criar uma Rede Neural do tipo LSTM

Com base na proposta do projeto em implementar uma rede neural que seja capaz de aprender através de inputs de forma contínua, faz-se necessário a habilidade da rede possuir certa memória através desses *inputs*, dando uma característica de aprendizado a partir de entradas anteriores, uma espécie de "memória" que possa recorrer para identificar de forma mais certeira dado a dados com características semelhantes.

A partir dessa necessidade de aprendizado o tipo de rede escolhida para esse projeto foi a rede neural do tipo LSTM (*Long Short Term Memory*), sendo está um modelo de rede que engloba de maneira sucinta a necessidade de guardar informações relevantes para serem utilizadas em treinamento. Além da rede LSTM, existem outros modelos de rede neural que são compatíveis com a abordagem de memória. Foi optado pela utilização da rede do tipo LSTM que foi implementada através de codificação na linguagem python.

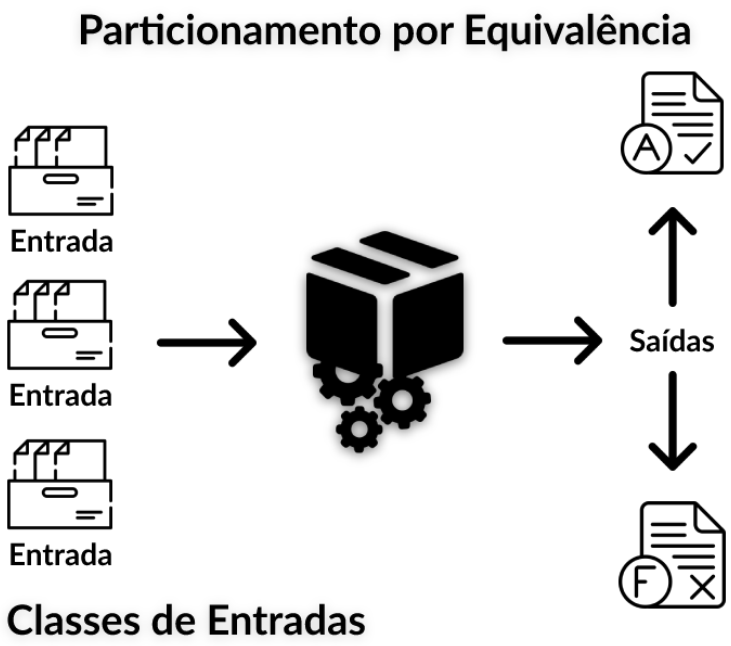
A rede foi então treinada com base no *dataset* de ocorrência de características contendo as áreas dos documentos de procedimentos jurídicos, mantendo sempre em sua camada *hidden* (sua memória), a identificação de áreas bem-sucedidas para reduzir o tempo de treino e o uso de recursos computacionais.

Foi utilizada a ferramenta moderna de código aberto amplamente utilizada no campo de *deep learning* chamada de TensorFlow, sendo essa um completo ecossistema voltado para desenvolvedores para a implementação de aplicações que usem da tecnologia de *machine learning* (TENSORFLOW CORE, 2021). Dentre as bibliotecas e ferramentas que o TensorFlow dispõe para uso, foi utilizada a API Keras, sendo está indicada na criação e treinamentos de modelos de *deep learning*. O Keras possui modelos e estruturas de camadas pré implementados para serem utilizados e moldados com a necessidade do usuário, para diminuir o *boilerplate* de implementar uma rede do zero em linguagens de baixo nível, sendo estas linguagens mais próximas do hardware, além do Keras oferecer compatibilidade com a linguagem de programação que será utilizada neste projeto.

3.2.4. Realização e descrição de Teste Funcional

Nesta etapa foi realizado e documentado o processo de teste de qualidade sobre a rede neural onde foi aplicado o teste funcional, também conhecido como teste de caixa preta. Nesse teste, o objetivo foi a validação das saídas da rede neural, que com o papel de identificar corretamente a área de atuação de processos extrajudiciais a partir de *n* entradas, independente de codificação assim o teste independe de cenário.

Figura 3: Particionamento por equivalência.



A técnica padrão que foi utilizada se chama particionamento por equivalência, como demonstrado na figura 3, onde foi realizada uma subdivisão de classes de entradas da rede por área de atuação, como é o objetivo de saída de previsão da rede. Uma sequência de *n* entradas válidas forma uma classe que se espera uma saída positiva, como o caso de identificação correta de área de atuação do procedimento extrajudicial. Uma segunda classe de *n* sequência de entradas inválidas para testar a saída e a invalidação da identificação, e por fim uma terceira classe de *n* sequência de entradas contendo entradas válidas e inválidas.

A documentação do processo consistiu na criação de arquivos de extensão *Markdown*, para facilitar estruturação e padronização sem abdicar de um conjunto de customizações e estilizações que a linguagem oferece. Os arquivos de documentação contaram com etapas de validação de entrada e saída adversos, onde foram testados cenários que não cabem ao propósito da rede em seu treinamento, para balancear o contraste na identificação de entrega de qualidade da rede, além de validar a saída e tentar identificar ou prever possíveis erros futuros.

4. RESULTADOS E DISCUSSÃO

Esta seção descreve os resultados da execução do trabalho sobre a rede neural, todo o processo para implementar e treinar a rede neural do tipo LSTM, dentro do contexto de identificação de área de atuação de documentos extrajudiciais, analisando e transformando os dados para serem usados na criação de um modelo de rede usando o Tensorflow e o Keras.

4.1. ANÁLISE DOS DOCUMENTOS

Na elaboração de um *dataset* compatível para com um modelo de rede neural utilizando o TensorFlow, é necessária a distribuição e classificação de metadados importantes para poder ser utilizada no treinamento da rede. Sabendo disso, foi realizada a análise de documentos para destacar pontos importantes, que podem ou não apontar uma diferença significativa em sua entrada para a rede neural.

A análise dessas características foi feita com base em um arquivo de formato *csv* compatível com o *Excel*, dentro do arquivo estavam inúmeras entradas de textos de documentos extrajudiciais que foram adquiridos do sistema do Ministério Público. O arquivo possui informações como: cabeçalho, tipo de solicitação do documento, partes envolvidas no requerimento, informações de arquivo de anexo, assinaturas das partes e informações de solicitações do requerente.

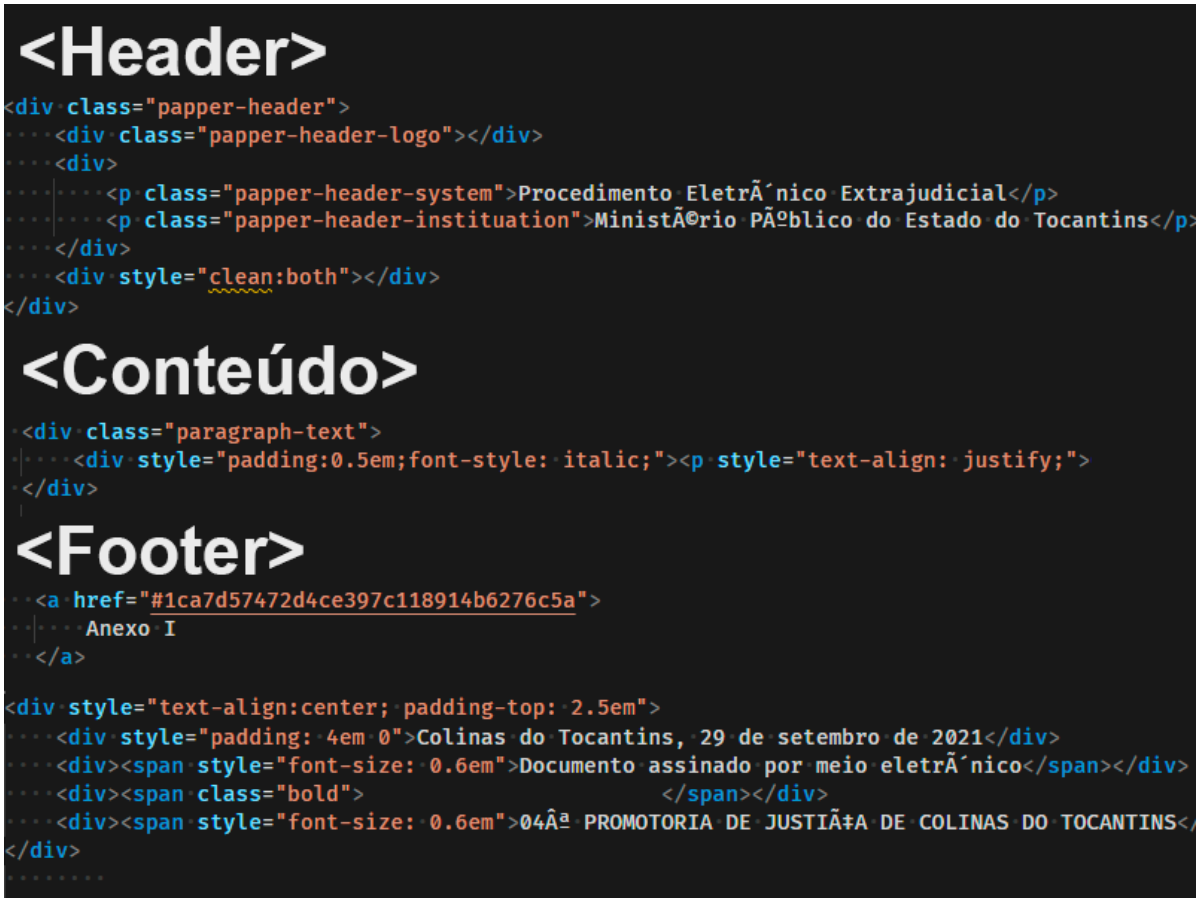
As colunas do arquivo original continham, em ordem: número, que continha o protocolo daquele documento; a área de atuação do documento, sendo este um dos 18 tipos de área de atuação, fundamental para entender os tipos possíveis de saída da rede; e, por fim, a última das três colunas continham o corpo do documento, sendo esta em formato HTML subdividido por diversas *tags* na linguagem de anotação de hipertexto, como demonstrado na figura 4.

Figura 4: Composição Arquivo CSV.

Para realizar o cruzamento de dados à procura das referências dentro do arquivo, foi necessário separar os dados de acordo com a área de atuação, como no caso da área do consumidor, onde foram extraídos do arquivo cerca de 380 entradas pertencentes a esta área de atuação. Tendo em mãos os dados específicos de uma área, foi realizada uma análise dos dados da coluna conteúdo, onde encontrava-se todo o texto relativo ao documento.

O conteúdo desta coluna possuía informações valiosas porque ela possuía dados como os requerentes do processo e as motivações que levaram a protocolização do documento, que são necessários para realizar o cruzamento destas palavras e apontar, ou não, a quantificação de sua notoriedade através dos distintos documentos extrajudiciais, assim desenvolvendo um *ranking* de palavras de alto índice de ocorrência.

Figura 5: Estruturação do Documento.



Como demonstrado na figura 5, as informações da coluna conteúdo podem ser subdivididas em 3 seções: Header, Conteúdo e Footer. A seção do Header, possui informações indicando que o documento é um procedimento extrajudicial pertencente ao Ministério Público do estado do Tocantins, essa informação foi uma das que, por não agregar valor à classificação como dado útil, não foi utilizada na extração de dados para compor o modelo de treinamento para a rede neural.

Na sequência, a seção do Conteúdo está em volta de uma *tag* na linguagem de marcação de hipertexto identificado como: *<p>*, que significa que toda informação encapsulada nela é considerada um parágrafo. Dentro desse parágrafo está todo o corpo do texto do documento, como a descrição do fato ocorrido que o requerente está apresentado, logo é o tipo de informação alvo para ser extraída como dado útil.

Por fim, a seção do Footer contém informações extras do documento, podendo conter ou não uma lista de informações de anexos do processo daquele determinado documento, sendo estas informações como o nome do arquivo anexado e o caminho de armazenamento daquele arquivo, além de também conter informações sobre os assinantes do documento e requerentes.

Esta seção também não se apresentou como dado útil no contexto de identificação de área de atuação do documento, por conter dados que não eram comumente recorrentes nos documentos, então ficou de fora no processo de extração.

4.2. Extração e Transformação dos Dados Para Cruzamento

Para processar o documento contendo as informações referentes aos textos dos procedimentos, foi criado um *script* escrito em *python*, que extrai os textos dentro da coluna de conteúdo na sua formatação original em HTML para um texto puro, como mostrado na figura 6.

Figura 6: Script para extração do conteúdo dos dados.

```
from bs4 import BeautifulSoup
import csv

arquivo = open('dados-docs-consumidor.csv', 'r')
ler_arquivo = csv.reader(arquivo, delimiter=';')

dados_extraidos = ''

for linha in ler_arquivo:
    conteudo = BeautifulSoup(linha[2])
    descricao_texto = conteudo.select('div.paragraph-text>div>p[style="text-align: justify;"]')
    for elemento in descricao_texto:
        texto = elemento.text
        if texto != []:
            dados_extraidos += texto + '\n'

arquivo.close()
arquivo_destivo = open('conteudo.txt', 'w')
arquivo_destivo.write(dados_extraidos)
arquivo_destivo.close()
```

O *script* utiliza-se da biblioteca csv, nativa do *python,* para ler arquivos do tipo planilha, juntamente com a biblioteca para processamento de arquivo hipertexto, o *BeautifulSoup*, sendo estauma ferramenta eficaz e poderosa na seleção e manipulação de arquivos estáticos de hipertexto extraídos da web (BEAUTIFUL SOUP, 2021).

O *script* primeiro abre e lê o arquivo de formato *csv*, que é transformado em um *array*, que é uma lista de itens, em que cada posição da lista corresponde a uma linha do arquivo. Após a criação da lista com base no arquivo, é feita uma iteração sobre seus itens. A cada iteração sobre a lista, são extraídas daquela linha, através do *BeautifulSoup*, as informações da coluna de conteúdo, apenas o texto. Após extrair o texto, é feita uma nova iteração ainda sobre a primeira, agora sobre o texto extraído. Nessa iteração é verificado se o texto extraído não é vazio, em caso de não ser, ele adiciona ao valor de saída final. Ao finalizar as iterações

sobre a lista, o valor de saída é guardado em um arquivo de formato *txt* que armazena o texto agora processado.

Esta etapa foi realizada para todos os diferentes 18 tipos de arquivos de planilha divididos em área de atuação, e foi importante para extrair apenas os textos que continham realmente algum conteúdo utilizável. Alguns arquivos apresentavam problemas com codificação de caracteres ou estavam corrompidos, o que os tornam inutilizáveis para um processamento correto das palavras. O que por sua vez acarretou uma diminuição drástica de amostras por categoria para serem trabalhadas.

4.3. Cruzamento dos Dados

O cruzamento dos dados foi a realização de contagem de ocorrência recorrente de palavras dentro dos arquivos *txt* gerados por categoria, na etapa de extração e transformação dos dados, contendo apenas os textos dos documentos extrajudiciais. Assim, foram verificadas as palavras que tinham maior ocorrência dentro das determinadas categorias, criando uma espécie de *ranking* de ocorrência. A figura 7 apresenta o *script* desenvolvido em *python* para realizar esse ranqueamento.

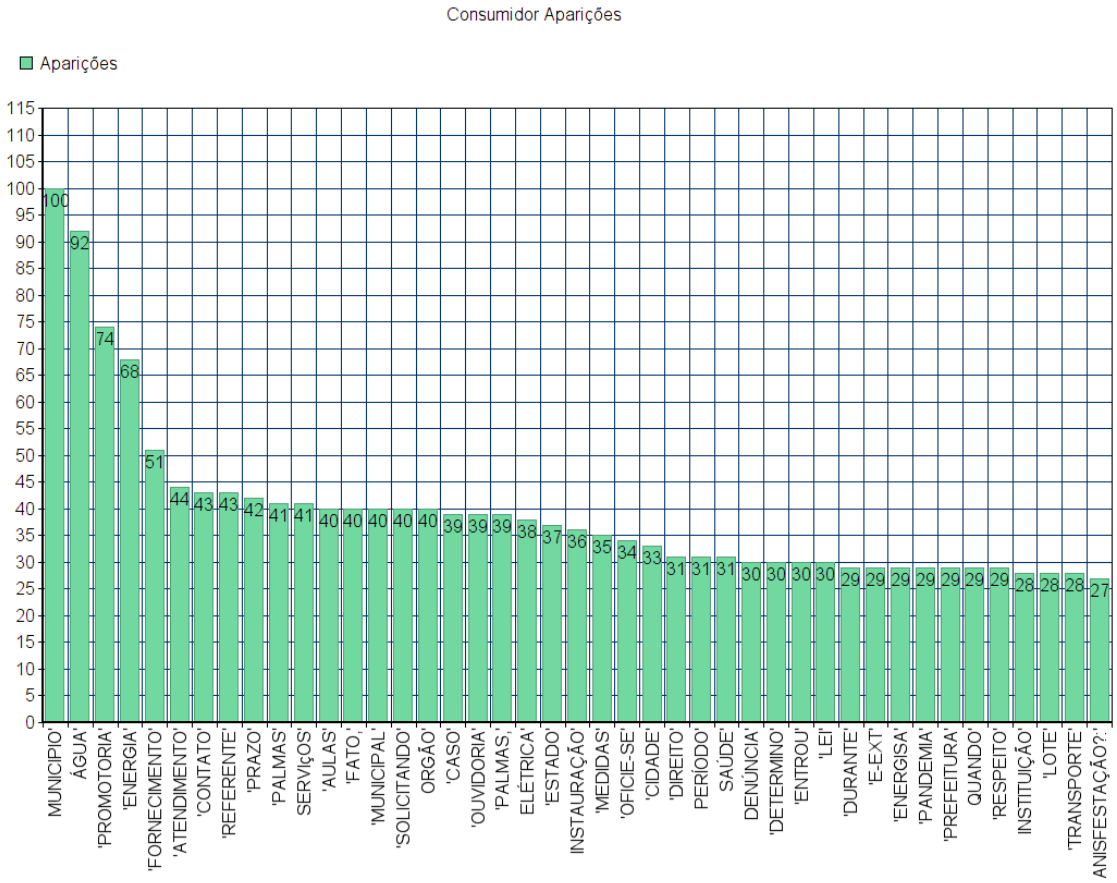
Figura 7: Algoritmo para contar ocorrências.

```
conteudo = open('conteudo.txt', 'r')
ler_conteudo = conteudo.read()
palavras = ler_conteudo.split(' ')
excluido = open('lista_exclusao.txt', 'r')
excluir = excluido.read()

rankeado = {}

for palavra in palavras:
    palavra = palavra.upper()
    if palavra not in excluir:
        if palavra not in rankeado:
            rankeado[palavra] = 1
        else:
            rankeado[palavra] += 1

conteudo.close()
excluido.close()
final = sorted([(value, key) for (key,value) in rankeado.items()])
```

O *script* contém como entrada o conteúdo dos arquivos de documentos extrajudiciais processados anteriormente e uma lista de exclusão, para remover palavras que não agreguem com o propósito do algoritmo como conectivos. Assim, é criado um dicionário em *python*, adicionando as ocorrências da palavra em caso da não existência do registro dentro do dicionário de exclusão, montando assim as palavras com maior índice de aparição aproveitável.

Por fim, o algoritmo dá um *sort* sobre o dicionário, para organizar os resultados em ordem decrescente, apresentando assim as palavras com maior ocorrência nos documentos. Como saída do processamento tem-se o gráfico apresentado na figura 8, demonstrando as palavras com maior ocorrência nos documentos extrajudiciais pertencentes à área do consumidor.

Figura 8: Aparições das palavras dentro da área de consumidor.



Como apresentado no gráfico da figura 8, as palavras com maior ocorrência dentro dos documentos da área de atuação do consumidor são: Município, Água, Promotoria e Energia. Esse processo em busca de palavras com ocorrência mais recorrente foi realizado em todas as 18 áreas de atuação, sendo que cerca de 4 dessas palavras principais por categoria foram selecionadas para comporem a abordagem da metodologia de entradas da rede neural.

4.4. CRIAÇÃO DO DATASET

A criação do *dataset* foi realizada a partir da abordagem de quantificação de ocorrências de palavras, então o formato da entrada da rede se dividiu na quantidade de ocorrências das palavras ranqueadas em cada categoria. Com o resultado do cruzamento dos dados da etapa anterior as categorias utilizadas para compor a rede estão ilustradas na figura 9.

Figura 9: Palavras ranqueadas geradas.

A figura 9 ilustra todas as palavras com maior ocorrência dentro de todas as 18 áreas de atuação dos documentos extrajudiciais. Com estas palavras foi mapeada a estrutura do *dataset*, sendo composta por área de atuação e as outras 80 palavras ranqueadas representando as *features*. Dentro de um *dataset* uma *feature*, também chamada de coluna, atributos ou

variáveis, é todo tipo de valor que pode ser medido, como quantidade de ocorrências de determinado atributo.

Para compor o *dataset* foi necessário quantificar as ocorrências dessas palavras dentro dos documentos extrajudiciais e guardá-los em uma estrutura de linhas e colunas. Foi utilizado o formato de arquivo de planilhas *csv*, por ter compatibilidade de transformação para entrada para treinamento de uma rede neural.

Para gerar a estrutura do *dataset* foi implementado um algoritmo em *python* para processar o arquivo de *csv* original, que não passou por quaisquer processamento ou transformação, por se tratar do arquivo com maior conteúdo. A estrutura do arquivo original consistia em linhas com valores distribuídos em colunas, sendo elas: número de protocolo, área de atuação e documento extrajudicial. A implementação do algoritmo em *python* pode ser vista na figura 10.

Figura 10: Script para gerar o dataset.

```python
def csv_adaptado(csvOriginal, features, savePath):
    print(f"{'#'*5} Iniciando a modelagem:\n")

    with open(csvOriginal, 'r') as csvFile:
        nome_arquivo = os.path.basename(csvFile.name)

        ler_csv = csv.reader(csvFile)

        with open(savePath, 'a', encoding='UTF8', newline='') as toSave:
            writer = csv.writer(toSave, delimiter=';')

            # Rodar para tentar encontrar ocorrencias do parâmetro, sobre cada row
            for row in ler_csv:
                # Instancia a lista com a area como o primeiro valor
                linha_nova = []
                #passando ao beautifulsoup para formatar os valores
                conteudo = BeautifulSoup(row[2])
                descricao_texto = conteudo.select(
                    'div.paragraph-text>div[style="padding:0.5em;font-style: italic;"]>p[style="text-align: justify;"]'
                )

                for texto in descricao_texto:
                    texto = texto.text.upper()

                    if texto != []:
                        linha_nova = [row[1].encode('iso-8859-1', 'replace').decode('utf-8', 'replace')]
                        for feature in features:
                            if texto.find(feature) == -1:
                                linha_nova.append(0)
                            else:
                                linha_nova.append(texto.count(feature))

                if linha_nova != []:
                    writer.writerow(linha_nova)
```

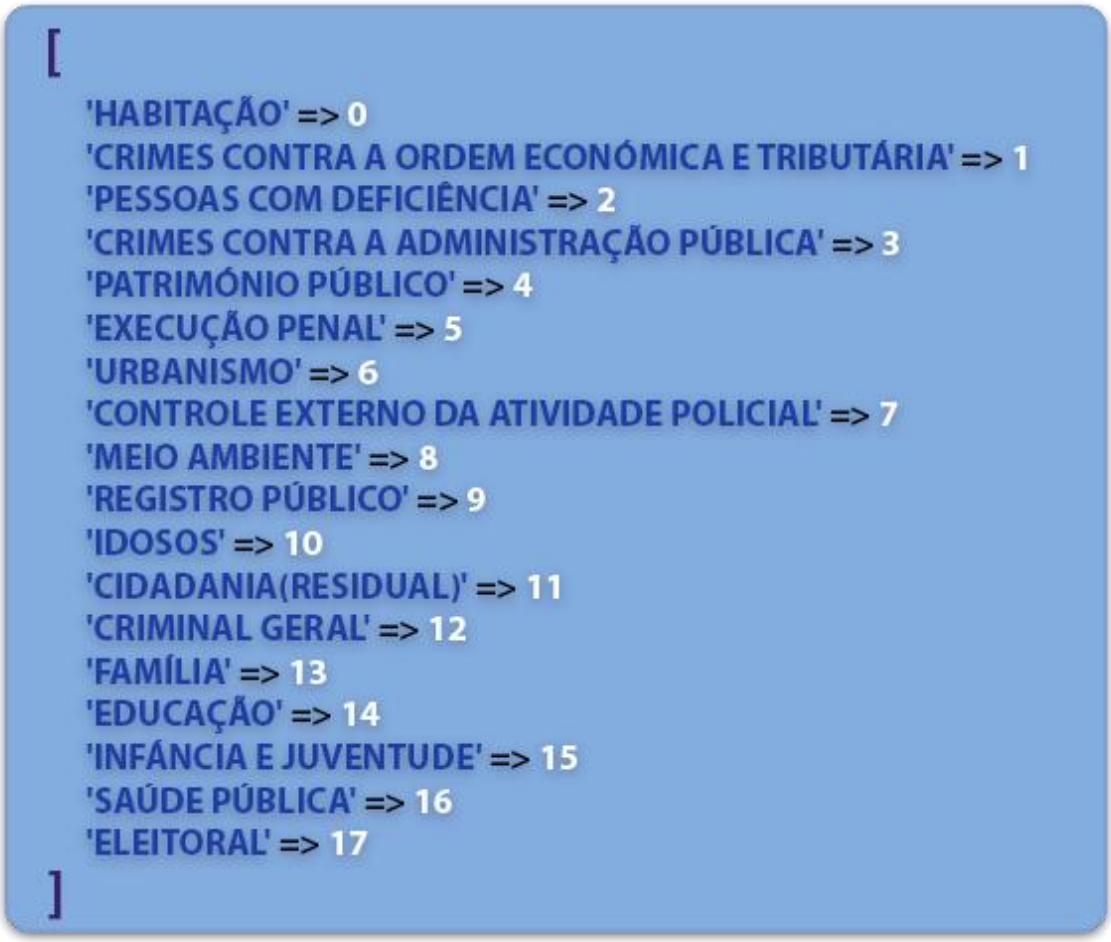
Como demonstrado na figura 10, o algoritmo recebe como parâmetros o arquivo *csv* original, contendo toda a estrutura já apresentada, as *features*, que são as 80 palavras ranqueadas, é um caminho para salvar o *csv* processado. Primeiro o algoritmo abre o arquivo e começa a iterar sobre as linhas dele, então utilizando-se da biblioteca do *BeautifulSoup* é feita uma extração do texto descritivo do requerente do documento extrajudicial.

Com os dados do documento, é feita uma iteração sobre as palavras do documento e comparado para checar a existência das palavras *feature*, em caso de ocorrência, e salva a quantidade de vezes que essa palavra ocorreu dentro do texto. Por fim, o documento é salvo seguindo a estrutura ilustrada na figura 11. O arquivo resultante *csv* ficou dividido entre a coluna da área de atuação e as 80 *features* quantificadas das palavras de ocorrência.

Figura 11: Divisão do dataset.

Como a rede neural aceita apenas valores inteiros e o tipo de saída esperada da rede é uma dentre 18 áreas de atuação possíveis, foi necessária a conversão dos valores descritivos das áreas de atuação que estavam em formato de texto para valores inteiros, convertidos para se enquadrar em uma *array* de 0 a 17, como apresentado na figura 12.

Figura 12: Áreas de atuação divididas na array.



Com todos os valores e *features* devidamente organizados e convertidos, o *dataset* foi finalizado, sendo ele feito de um arquivo csv, composto por uma coluna de área de atuação, que será o alvo na identificação que a rede neural irá tentar prever, e mais 80 features que serviram para a rede neural tentar aprender e distinguir as 18 áreas de atuação.

4.5. Criação da Rede Neural LSTM

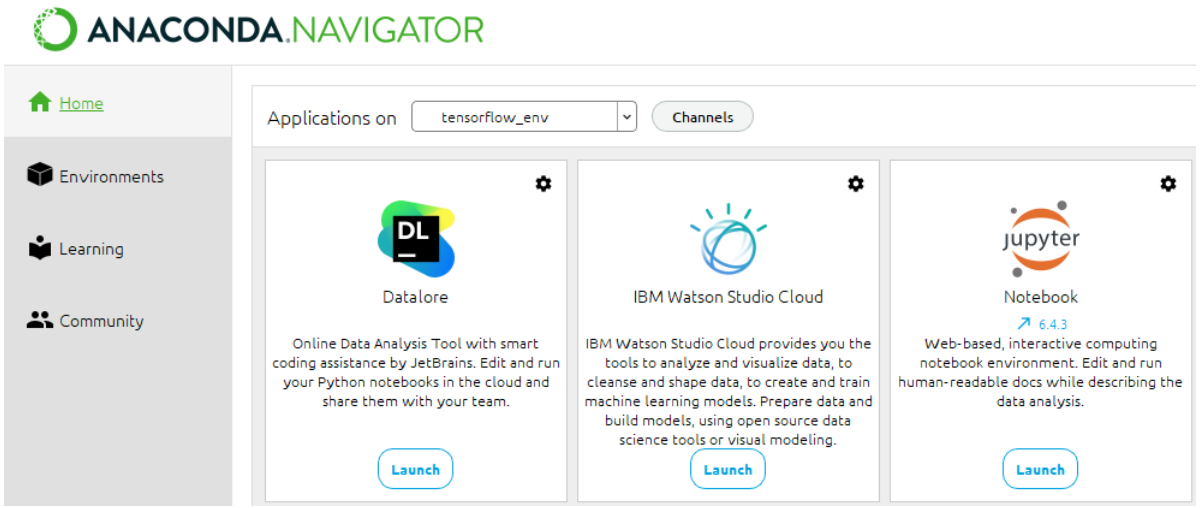
A criação da rede neural foi separada em duas etapas: a configuração do ambiente de desenvolvimento, usando a distribuição Anaconda, e a criação do modelo de rede neural utilizado o TensorFlow e a API do Keras, na construção de um modelo de rede neural que atendesse o propósito deste trabalho, sendo que o tipo de rede neural escolhido foi o LSTM.

4.5.1. Configuração do ambiente

O início do desenvolvimento da rede neural se deu na criação e configuração do ambiente da rede neural utilizando a distribuição Anaconda, que é uma ferramenta para gerir e organizar

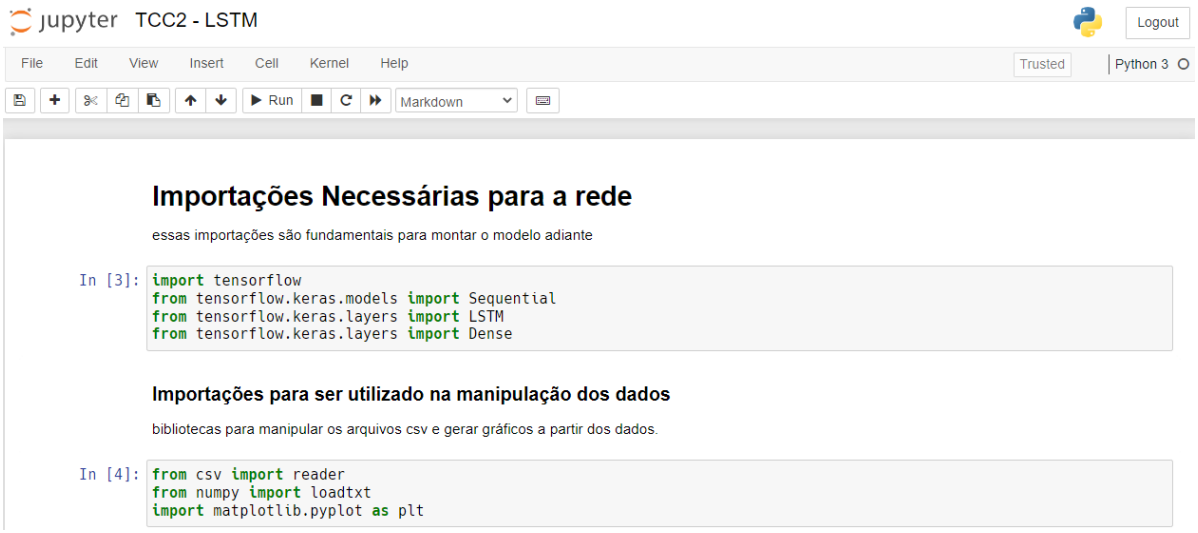
módulos e dependências de forma gratuita. Utilizando da ferramenta Anaconda, o *Navigator* foi criado um *environment*, que é um ambiente configurado apenas com as bibliotecas necessárias para a implementação desse projeto. O ambiente criado foi chamado de *tensorflow_env*, como mostra a figura 13.

Figura 13: Anaconda com o ambiente Tensorflow configurado.



Dentro desse ambiente foi criada uma instância do Jupyter Notebook, que é uma espécie de caderno de anotação para auxiliar no desenvolvimento e organização de código. Ao utilizar o Jupyter como ambiente para armazenar, rodar e organizar os códigos, foi possível criar uma base de código propriamente documentada. Onde foi possível criar células anotadas indicando o uso de determinados segmentos e trechos do código, utilizando sintaxe de linguagem Markdown, uma linguagem própria para anotação, como mostra a figura 14.

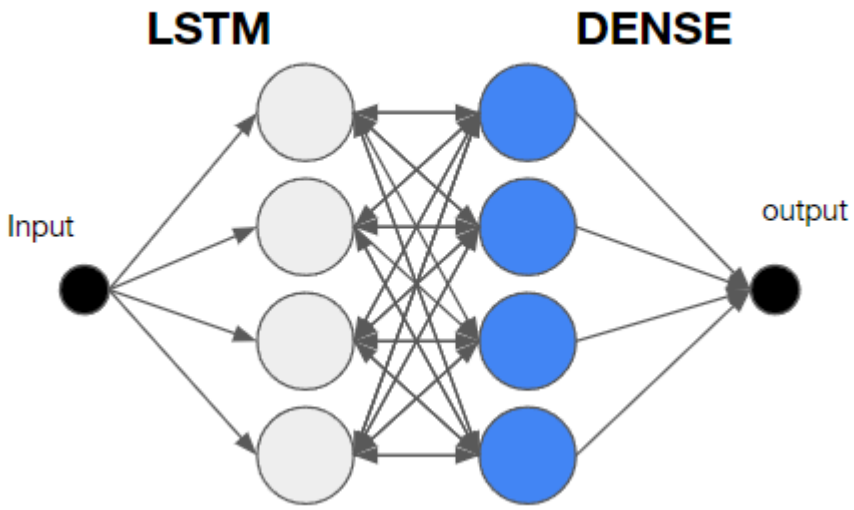
Figura 14: Jupyter Notebook a base de armazenamento do código.



4.5.2. Criação do Modelo de Rede Neural

Na etapa de criação do modelo de rede neural, foi realizada a composição das camadas que a rede neural iria possuir. Como demonstrado na figura 15, a rede ficou estruturado em uma camada de *input*, uma camada LSTM, uma camada do tipo DENSE e uma do tipo *output*.

Figura 15: Estrutura da rede neural



Inicialmente, a rede é instanciada criando um modelo do tipo *Sequential*, sendo que este modelo possibilitou a inserção de camadas em sequência, fazendo com que o *output*, a saída que a camada processa do *input*, seja utilizada como input da camada seguinte.

O modelo criado englobado pelo Sequential, possui como camada de entrada a camada LSTM, uma camada do tipo densa. A camada LSTM recebe os parâmetros de quantidade de neurônios, também chamados de células. No modelo atual foi optado por criar uma camada contendo 1024 neurônios. Quanto maior o número de neurônios melhor a rede neural consegue aprender dados complexos, entretanto, isso vem ao custo do desempenho pois vai ser requerido um maior poder de processamento para treinar a rede, tornando o processo mais lento.

O segundo parâmetro para instanciar a camada LSTM é o *return_sequences*, para o qual foi passado o valor *true* para indicar que a camada LSTM retorne o estado dela a cada iteração. E, por fim, o último parâmetro passado foi o *input_shape*, que é onde fica o formato da entrada que a rede vai receber, e como foram trabalhadas linhas com 80 *features*, esse será o formato de *input* que a rede espera receber.

A segunda camada do modelo, a camada densa, recebe input de todos os neurônios da camada LSTM, no total de 18 neurônios, sendo essa a quantidade de saídas possíveis de classificação de áreas de atuação. Esta camada densa contou com um método de ativação, passado pelo parâmetro *activation*. As funções de ativação são responsáveis por definir como o peso da soma do *input* é transformado em um *output* de um ou mais neurônios.

Como o problema a ser solucionado foi a identificação de diferentes tipos de áreas de atuação de documentos extrajudiciais, o tipo de *activation* que a camada densa recebeu foi o *softmax*. O *softmax* é utilizado em problemas de classificação de múltiplas classes, quando o problema não é resolvido com classificação binária. Este tipo de ativação retorna dependendo da quantidade de neurônios, uma *array* com os prováveis resultados. O modelo implementado da rede neural pode ser visto na figura 16.

Figura 16: Criação do modelo da rede neural.

Montando o modelo da Rede Neural

o modelo é instanciado a partir da classe Sequential.

São adicionados as camadas LSTM e Dense.

```
model = Sequential()
model.add(LSTM(1028, return_sequences=True, input_shape=(1, 80)))
model.add(Dense(18, activation='softmax'))
```

Com o modelo criado foi necessário realizar a compilação do modelo, para poder validar se o modelo criado é realmente um modelo de rede neural válido. Para isso, foi necessário passar 3 parâmetros, sendo eles o *optmizer*, que é a função que irá dizer como os pesos da rede neural serão atualizados; o *loss*, que conta com a função de comparação com os dados de validação; e o *metrics*, aquilo que irá ser guardado para avaliação. A figura 17 apresenta os parâmetros passados para o método compile.

Figura 17: Compilando o modelo da rede neural.

Compilando o modelo da rede neural

```
model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Foi utilizado como *optmizer* o *rmsprop*, bem popular dentro da área de aprendizado de máquina, com uma abordagem de gradiente descendente, isto é, encontrar os parâmetros que minimizam as funções de maneira iterativa. Como função *loss*, foi utilizado o método *sparse_categorical_crossentropy*, este método é usado quando o *output* da rede pode ser vários tipos de saídas, chamado de *multi-class classification*, assim ele produz o *index* mais provável que vá corresponder a determinada categoria. E, por último, foi apontada a métrica *accurary*, para ser usada como avaliação do desempenho dos treinamentos e validações.

Com o modelo já pronto para receber as entradas para o treinamento, foi importado no ambiente do Jupyter Notebook, o *dataset* que foi preparado para a rede, foram realizados ajustes para se adequar ao input aceito pela rede neural, como demonstrado na figura 18, e foi realizada uma transformação dos dados.

Figura 18: Carregando o dataset.

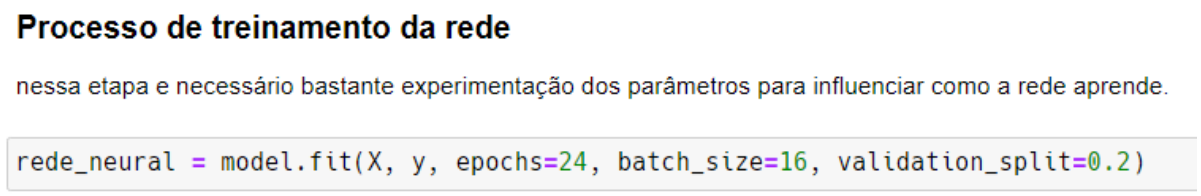
Como apresentado na figura 18, o *dataset* foi separado nas variáveis *X* e *y*, sendo o conteúdo da variável *X* as entradas que serão utilizadas no treinamento e validação da rede e *y* o alvo da rede neural, os valores da primeira coluna do *dataset*, as áreas de atuação.

Foi necessário realizar um *reshape* sobre o *dataset* em X, já que as redes neurais do tipo LSTM recebem apenas valores de entrada em 3 dimensões, e originalmente por se tratar de um arquivo *csv*, ele possuía apenas duas dimensões. Assim, o shape, que é a forma dos dados, ficou organizado com 7118 entradas, sendo a entrada composta por uma linha e 80 *features*.

Já com o modelo compilado e o *dataset* importado, foi iniciado o processo de experimentação com parâmetros de treinamento da rede. Para isso, foram necessários ajustes de quantidade de amostras aplicadas, a quantidades de *epochs* a serem processadas e

quantidade de amostras que seriam utilizadas para a validação da rede, como demonstra a figura 19 com os valores mais otimizados para este projeto de rede.

Figura 19: Processo do treinamento da rede neural.



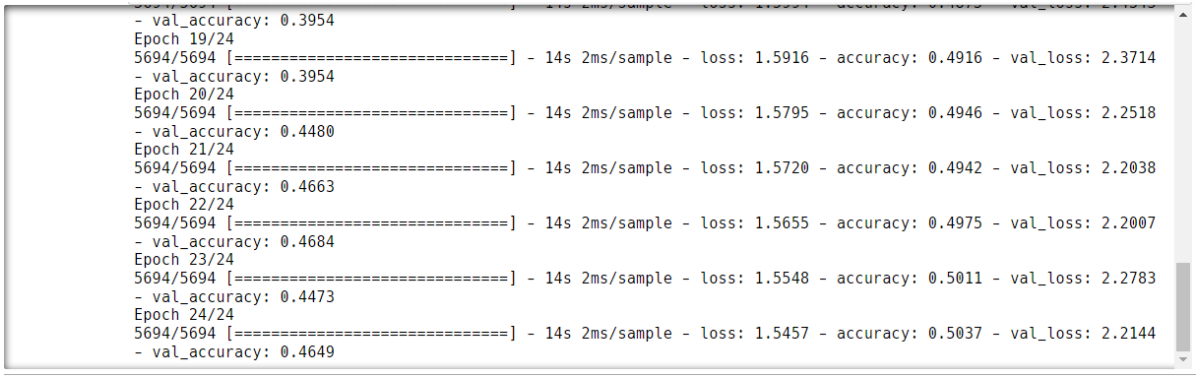
O método *fit* recebe o *dataset* a ser processado representado por *X*, junto aos valores alvos representados por *y*, juntamente a quantidade de *epochs*, que são a quantidade de vezes que será iterado sobre o dataset, o *batch_size*, que é a quantidade de entradas que é processado antes que o modelo seja atualizado, e, por fim, o *validation_split*, que a quantidade de entradas que fica reservado para ser utilizada como validação para comparação.

Após fazer diversas experimentações nos parâmetros para o treinamento da rede neural, a combinação de parâmetros que mostrou os resultados mais otimizados para o contexto de identificação da área de atuação com as menores perdas possíveis foi a de 24 *epochs*, logo um total de 24 gerações de iteração sobre o dataset. Um *batch_size* de tamanho 16 para que a rede pudesse aprender de forma espaçada, é um *validation_split* de 0.2, sendo assim dedicando cerca de 0.2 dos *inputs* do *dataset* para servirem de validação.

Quando o processo de treino da rede é iniciado, é possível acompanhar em tempo real o progresso de aprendizado da rede neural. São exibidas informações do progresso da rede como a *epoch* atual, a porcentagem de *inputs* percorridos, e a quantidade de *loss*, que nos indica o quão bem a rede está aprendendo com as entradas, quanto mais próximo de 0, que é o caso ideal, melhor a rede está absorvendo o aprendizado.

Outros parâmetros que foram observados no decorrer do treino foi o de *accuracy*, que demonstrou a porcentagem do quão certa a rede está prevendo os *labels* que são os *outputs* a serem previstas, as áreas de atuação que foram armazenados na variável *y*. E, por último, o parâmetro *val_loss* indica o quanto a rede aprendeu naquela *epoch* em comparação com a anterior. É possível observar as saídas de treinamento como demonstra a figura 20.

Figura 20: Saída do treinamento da rede neural.



```
- val_accuracy: 0.3954
Epoch 19/24
5694/5694 [==============================] - 14s 2ms/sample - loss: 1.5916 - accuracy: 0.4916 - val_loss: 2.3714
- val_accuracy: 0.3954
Epoch 20/24
5694/5694 [==============================] - 14s 2ms/sample - loss: 1.5795 - accuracy: 0.4946 - val_loss: 2.2518
- val_accuracy: 0.4480
Epoch 21/24
5694/5694 [==============================] - 14s 2ms/sample - loss: 1.5720 - accuracy: 0.4942 - val_loss: 2.2038
- val_accuracy: 0.4663
Epoch 22/24
5694/5694 [==============================] - 14s 2ms/sample - loss: 1.5655 - accuracy: 0.4975 - val_loss: 2.2007
- val_accuracy: 0.4684
Epoch 23/24
5694/5694 [==============================] - 14s 2ms/sample - loss: 1.5548 - accuracy: 0.5011 - val_loss: 2.2783
- val_accuracy: 0.4473
Epoch 24/24
5694/5694 [==============================] - 14s 2ms/sample - loss: 1.5457 - accuracy: 0.5037 - val_loss: 2.2144
- val_accuracy: 0.4649
```

Com a rede neural treinada, foi necessário fazer uma avaliação para checar se a rede era adequada para o problema de identificar a área de atuação de documentos extrajudiciais, dada a abordagem por quantificação. Além disso, o processo de avaliação também verificou se o *dataset* é realmente adequado para o treino. Como observado na figura 21.

Figura 21: Accuracy do modelo da rede neural.

```
# avaliar o modelo da rede neural
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

```
7118/7118 [==============================] - 1s 190us/sample - loss: 1.6542 - accuracy: 0.5032
Accuracy: 50.32
```

O modelo contou com uma *accuracy* de 50.32%, o que indica um caso de *underfitting* na rede neural. A rede neural desenvolvida pode prever as áreas de atuação, mas a *accuracy* apresentada pode ou não levar a uma identificação não prevista em grande maioria dos casos.

Os fatores que levam uma rede neural a ser classificada como *underfitting*, são variados, como os dados de *input* serem generalizados demais o que deixa pouco espaço para distinção, ou até mesmo uma baixa performance de treino. Outro fator que pode impactar é a quantidade de *inputs* que se demonstram insuficientes para a rede conseguir formar algum tipo de distinção de resultados.

A rede neural sofreu com o caso em que seus *inputs* não se demonstraram suficientemente distintos, seja por baixa quantidade de entradas distintas entre cada área de atuação ou por baixo índice de ocorrência dentre as colunas de *feature* que o *dataset* possuía.

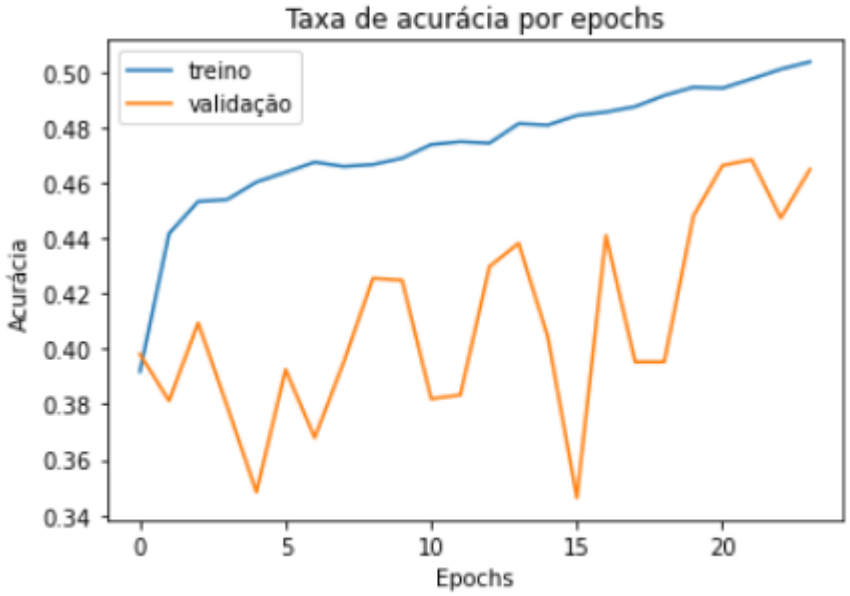
Para extrair e visualizar os dados de validação do treinamento de rede, foi utilizado a biblioteca *matplotlyb*, mais especificamente o módulo *pyplot*, para gerar gráficos para

acompanhar a evolução da aprendizagem da rede a cada *epoch*, e pode ser observado na figura 22.

Figura 22: Gráfico de acompanhamento de taxa de acurácia da rede.

```
plt.plot(rede_neural.history['accuracy'])
plt.plot(rede_neural.history['val_accuracy'])
plt.title('Taxa de acurácia por epochs')
plt.xlabel('Epochs')
plt.ylabel('Acurácia')
plt.legend(['treino','validação'])
```

```
<matplotlib.legend.Legend at 0x13c9fabba58>
```



A figura 22 apresenta os resultados sobre o treinamento da rede neural, onde foi denotado que a rede teve uma diferença de 0.04% entre os dados de treinamento e de validação, em outros casos tendo uma margem bem maior com o passar das *epochs*, o que indicou um *gap* entre a aprendizagem do modelo. Esse fenômeno pôde ser mais bem observado quando foi gerado o gráfico demonstrando a quantidade de perda de aprendizagem que rede teve com o passar das *epochs*, como demonstrado na figura 23, onde vale notar que quanto mais próximo de 0 melhor para o caso de perda, onde neste caso os resultados foram ruins.

Figura 23: Gráfico com a perda por épocas da rede.

```
plt.plot(rede_neural.history['loss'])
plt.plot(rede_neural.history['val_loss'])
plt.title('Perda por épocas')
plt.xlabel('Épocas')
plt.ylabel('Perda')
plt.legend(['treino','validação'])
```

```
<matplotlib.legend.Legend at 0x13ce4c717b8>
```

4.5.3. Teste da Rede Neural de Caixa Preta

Os testes sobre a rede neural foram escolhidos para serem seguidos de acordo com a metodologia de testes de caixa preta, onde não se era conhecido o conteúdo das entradas ou saídas, então foi montado um *dataset* para servir como input seguindo o shape de entrada aceito pela rede neural, 1 linha com 80 *features*. Essas entradas então foram dadas à rede e foi

feita uma validação para verificar se a saída correspondia com os valores esperados, como demonstrado na figura 24.

Figura 24: Carregando entradas para o teste de caixa preta.

Com o *dataset* de testes carregado, foi feita a moldagem do shape de entrada para corresponder à entrada que o modelo de rede neural espera receber, assim o *dataset* passou a possuir 3 dimensões. Para poder visualizar a saída dos resultados da rede de forma mais interativa, foi criado um *array* contendo as identificações das áreas de atuação, pois a rede neural vai entregar como saída uma *array* de posições prováveis, pois foi utilizado no modelo

da rede neural o método de *activation softmax* na camada densa de saída. O *array* criada pode ser observada na figura 25.

Figura 25: Array de saídas esperadas pela rede neural.

Array com as possíveis saídas

```
area_atuacao = ['HABITAÇÃO',
'CRIMES CONTRA A ORDEM ECONÓMICA E TRIBUTÁRIA',
'PESSOAS COM DEFICIÊNCIA',
'CRIMES CONTRA A ADMINISTRAÇÃO PÚBLICA',
'PATRIMÓNIO PÚBLICO',
'EXECUÇÃO PENAL',
'URBANISMO',
'CONTROLE EXTERNO DA ATIVIDADE POLICIAL',
'MEIO AMBIENTE',
'REGISTRO PÚBLICO',
'IDOSOS',
'CIDADANIA(RESIDUAL)',
'CRIMINAL GERAL',
'FAMÍLIA',
'EDUCAÇÃO',
'INFÂNCIA E JUVENTUDE',
'SAÚDE PÚBLICA',
'ELEITORAL']
```

Para fazer a predição, foi utilizado o método *predict* que, por ser tratar de um modelo do Keras, já possuía o método para realizar a previsão dos dados inseridos, o método recebeu o *dataset* de teste para verificar se ele conseguiria prever corretamente as saídas com o modelo já treinado. As entradas de testes consistiam em 31 inputs com 80 *features*, sendo que elas já possuíam uma área de atuação vinculada a elas. O código feito para iterar sobre os resultados

pode ser visto na figura 26, foi utilizada a biblioteca do *numpy* para poder pegar o maior valor da saída da previsão, já que a rede neural retorna um array de possibilidades.

Figura 26: Código para iterar sobre os resultados da rede.

```
predicao = model.predict(inputs)

for md in range(31):
    print('Previsão da Rede Neural:', area_atuacao[np.argmax(predicao[md])])
    print('Deveria ser:', area_atuacao[np.int(outputs_esperados[md])])
    print('#######')
```

Os resultados da predição não foram completamente satisfatórios, na maior parte das tentativas de predição a rede neural não conseguiu prever a área de atuação correta, mas em outras áreas ela conseguiu prever corretamente, o que apontou uma discrepância entre as *features* que estavam associadas à área de atuação. A área de crimes contra a ordem tributária

foi a área que a rede conseguiu prever na maioria dos casos e a área de habitação foi a que ela não conseguiu prever com precisão. As saídas de previsão podem ser observadas na figura 27.

Figura 27: Saídas da previsão da rede neural.

```
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: CRIMINAL GERAL
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: MEIO AMBIENTE
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: HABITAÇÃO
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: MEIO AMBIENTE
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: PATRIMÔNIO PÚBLICO
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
Previsão da Rede Neural: CRIMINAL GERAL
Deveria ser: CRIMES CONTRA A ORDEM ECONÔMICA E TRIBUTÁRIA
#######
```

Com isso, o teste de caixa preta se demonstrou eficaz para poder validar o fator de acerto das predições do modelo da rede neural, onde foi utilizado um *dataset* a parte com o formato aceito pela rede e todas as suas *features* em busca de uma predição correta.

5. CONSIDERAÇÕES FINAIS

O presente trabalho descreveu o desenvolvimento de uma rede neural do tipo LSTM, com o objetivo de identificar áreas de atuação de documentos extrajudiciais, utilizando o Tensorflow e a API do Keras.

O desenvolvimento da rede neural englobou a criação de um *dataset* de entradas compatíveis com uma rede neural que foram extraídas de documentos de procedimentos extrajudiciais, onde foi necessária a estruturação e adequação dos dados por virem de fontes externas e fora do controle deste projeto. Os dados iniciais foram apresentados em um formato não estruturado e com problemas de codificação de caracteres, o que impactou a abordagem utilizada para a composição do *dataset* final.

Devido a necessidade de tratamento e processamento dos dados, grande parte das amostras do arquivo *csv* original foram perdidas, o que se deu pelos problemas com codificação de caracteres e diferentes representação de palavras como erros gramaticais dentro dos documentos. Isso acarretou um *dataset* de tamanho reduzido, o que influenciou na baixa taxa de sucesso de treinamento da rede neural. Assim, foi denotada a desigualdade na distribuição de entradas de determinadas áreas, onde algumas áreas de atuação presentes no *dataset* possuíam pouca quantidade de entradas com relação a outras áreas.

O modelo desenvolvido neste trabalho foi composto por uma camada LSTM com objetivo de armazenar informações de *inputs* da rede para que ela aprendesse, fazendo assim diminuir o tempo de treinamento da rede em busca de contornar a limitação do *dataset*. E, uma camada de saída do tipo densa que fazia conexão com todos os neurônios da camada de entrada LSTM, a camada densa dava um output contendo um array com um dos prováveis 18 tipos de áreas de atuação.

A rede neural modelada contou com uma baixa performance de treinamento e validação, o que a caracterizou como imprópria para o uso efetivo em seu estado atual. Neste trabalho, a rede neural desenvolvida cumpre com o objetivo de identificar as áreas de atuação dos documentos extrajudiciais, porém com uma taxa de sucesso de identificação de 50,32%.

Para trabalhos futuros, propõem-se a busca por outra abordagem para interpretar os dados recebidos do Ministério Público do estado do Tocantins, e realizar uma padronização mais profunda dos documentos extrajudiciais, desde a formatação dos documentos à codificação do documento até a escrita correta das palavras, para favorecer o processamento dos documentos para serem utilizados em uma rede neural. Utilizar PLN para identificar entidades nomeadas nos textos dos documentos para identificar com maior precisão as partes dentro do texto. Com o objetivo de identificar características que possam ser utilizadas na

composição de um *dataset* mais rico em diversidade de *inputs* entre diferentes tipos de área de atuação.

6. REFERÊNCIAS

Beautiful Soup**. Beautiful Soup Documentation** - Beautiful Soup 4.9.0 documentation. Disponível em: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Acesso em: 02/10/2021.

BRASIL. Constituição (1988). **DOS DIREITOS E DEVERES INDIVIDUAIS E COLETIVOS**. Art 5. Brasília, DF: Senado Federal: Centro Gráfico, 1988.

BRASIL. Lei nº 11.419. **Processo Eletrônico – Lei nº 11.419, de 19 de dezembro de 2006**. Presidência da República Casa Civil: Subchefia para Assuntos Jurídicos, Brasília, DF, 19 out. 2006.

BUDIANSKY, Stephen. **Lost in Translation**. Atlantic Monthly, 1998.

FUKUSHIMA, Kunihiko; MIYAKE, Sei; ITO, Takayuki. Neocognitron: A neural network model for a mechanism of visual pattern recognition. **IEEE transactions on systems, man, and cybernetics**, n. 5, p. 826-834, 1983.

GRAVES, Alex. *Long short-term memory*. In: ***Supervised sequence labelling with recurrent neural networks*.** Springer, Berlin, Heidelberg, 2012. p. 37-45.

HAYKIN, Simon. **Redes neurais: princípios e prática**. Bookman Editora, 2007.

HIRSCHBERG, Julia; MANNING, Christopher D. Advances in natural language processing. **Science**, v. 349, n. 6245, p. 261-266, 2015.

JOSHI, Aravind K. Natural language processing. **Science**, v. 253, n. 5025, p. 1242-1249, 1991.

JURASKY, Daniel; MARTIN, James H. Speech and Language Processing: An introduction to natural language Processing. **Computational Linguistics and Speech Recognition. Prentice Hall, New Jersey**, 2000.

RAU, Lisa F. Extracting company names from text. In: **Proceedings the Seventh IEEE Conference on Artificial Intelligence Application**. IEEE Computer Society, 1991. p. 29, 30, 31, 32-29, 30, 31, 32.

REITER, Ehud and DALE, Robert. **Building Natural Language Generation Systems**. Cambridge University Press, 2000. Natural Language Engineering, v. 7, n. 3, p. 271, 2001.

McCulloch, Warren S., and Walter Pitts. **"A logical calculus of the ideas immanent in nervous activity."** *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

MOHIT, Behrang. Named entity recognition. In: **Natural language processing of semitic languages**. Springer, Berlin, Heidelberg, 2014. p. 221-245.

MEDSKER, Larry R.; JAIN, L. C. **Recurrent neural networks**. Design and Applications, v. 5, 2001.

O'REGAN, Gerard. Marvin Minsky. In: **Giants of Computing**. Springer, London, 2013. p. 193-195.

PROCURADORIA-GERAL DE JUSTIÇA (Tocantins). PROCURADORIA-GERAL DE JUSTIÇA. Ato nº 030/2016, de 15 de abril de 2016. Regulamentação Extrajudicial Definitivo. **Ato N° 030/2016:** Institui e Regulamenta o Programa Eletrônico de Registro, Acompanhamento e Organização das Atividades Finalísticas Extrajudiciais do Ministério Público do Estado do Tocantins., Palmas, ano 16, n. 30, p. 1-11, 15 abr. 2016. Disponível em: https://athenas.mpto.mp.br/athenas/FileUploadController/get_public_file/0f85619bfeb09dbd6ea776b0f5bd13cb. Acesso em: 19 abr. 2021.

SHERSTINSKY, Alex. **Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network**. Physica D: Nonlinear Phenomena, v. 404, p. 132306, 2020.

TensorFlow Core. **Tensorflow**: Core | Machine learning. Disponível em: https://keras.io/. Acesso em: 28 julho 2021.

WANG, Sun-Chong. Artificial neural network. In: **Interdisciplinary computing in java programming**. Springer, Boston, MA, 2003. p. 81-100.

WERBOS, Paul John. **The roots of backpropagation: from ordered derivatives to neural networks and political forecasting**. John Wiley & Sons, 1994.

WERBOS, Paul J. Generalization of backpropagation with application to a recurrent gas market model. **Neural networks**, v. 1, n. 4, p. 339-356, 1988.