**Universidade de Brasília**
**Faculdade de Tecnologia**

# Aprendizado Profundo Aplicado a Classificação de Textos Longos com Poucos Dados: O Caso do PPF

Carlos Alberto Alvares Rocha

DISSERTAÇÃO DE MESTRADO

SISTEMAS MECATRÔNICOS

Brasília

2023

**Universidade de Brasília**
**Faculdade de Tecnologia**

# Aprendizado Profundo Aplicado a Classificação de Textos Longos com Poucos Dados: O Caso do PPF

Carlos Alberto Alvares Rocha

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS

Orientador: Prof. Dr. Li Weigang

Brasília
2023

Alberto Alvares Rocha, Carlos.
A769a Aprendizado Profundo Aplicado a Classificação de Textos Longos com Poucos Dados: O Caso do PPF / Carlos Alberto Alvares Rocha; orientador Li Weigang. -- Brasília, 2023.
126 p.

Dissertação de Mestrado (Sistemas Mecatrônicos) -- Universidade de Brasília, 2023.

1. Palavra chave 1. 2. Palavra chave 2. 3. Palavra chave 3. 4. Palavra chave 4. I. Weigang, Li, orient. II. Título

**Universidade de Brasília**
**Faculdade de Tecnologia**

# Aprendizado Profundo Aplicado a Classificação de Textos Longos com Poucos Dados: O Caso do PPF

Carlos Alberto Alvares Rocha

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS

Trabalho aprovado. Brasília, 16 de março de 2023:

---

**Prof. Dr. Li Weigang, UnB/IE/CIC**
Orientador

---

**Prof. Dr. Victor Rafael Rezende Celestino, UnB/FACE/ADM**
Membro interno

---

**Prof. Dr. Marcelo Xavier Guterres, ITA**
Membro externo

---

**Prof. Dr. Daniel Mauricio Munoz Arboleda, UnB/FT/ENM**
Membro suplente

Brasília
2023

*Este trabalho é dedicado a você, familiar ou amigo*
*que contribuiu muito na minha caminhada.*

# Agradecimentos

Ao grupo de pesquisa, por todas as contribuições e direcionamentos durante o desenvolvimento do projeto. Ao MCTI, pela parceria com a universidade e apoio financeiro. Ao meu estimado orientador Li Weigang, que me ajudou com o seu apoio, conhecimento e orientações. À todos os professores e servidores do PPMEC, pelo suporte e ensinamentos. Aos meus queridos amigos, pelo apoio demonstrado ao longo de todo o período dedicado a este trabalho, em especial à minha amiga Rafaela Sinhoroto, pelo apoio no desenvolvimento dos códigos. À minha namorada Luiza, pelo companheirismo e por toda a ajuda na revisão deste documento. Aos meus pais, meus maiores incentivadores, pela educação que me deram e pelo amor que me dedicam.

*“Yes, there were times*
*I’m sure you knew*
*When I bit off*
*More than I could chew*
*But through it all*
*When there was doubt*
*I ate it up and spit it out*
*I faced it all and I stood tall*
*And did it my way”*
*(Frank Sinatra)*

# Resumo

O processamento de linguagem natural (PLN) é uma área da inteligência artificial que vem ganhando bastante atenção nos últimos anos. Os grandes avanços recentes atraíram a atenção do Ministério de Ciência, Tecnologia e Inovações (MCTI) para a execução de um projeto com o objetivo de localizar financiamento internacional para pesquisa e desenvolvimento acessível aos pesquisadores brasileiros por meio do seu Portfólio de Produtos Financeiros (PPF). A classificação aparece como um desafio para essa solução devido a ausência de dados rotulados de alta qualidade, que são requisitos para a maioria das implementações estado-da-arte na área. Este trabalho explora diferentes estratégias de aprendizado de máquina para classificar os textos longos, não estruturados e irregulares, obtidos através da raspagem de websites de instituições de financiamento, para, através de uma abordagem incremental, encontrar um método adequado com um bom desempenho. Devido a limitada quantidade de dados disponível para o treinamento supervisionado, foram empregadas soluções de pré-treinamento para aprender o contexto das palavras a partir de outros conjuntos de dados, com grande similaridade e maior tamanho. Em seguida, utilizando as informações adquiridas, foi aplicada uma transferência de aprendizado associada a modelos de aprendizado profundo para aprimorar a compreensão de cada sentença. Para diminuir o impacto da irregularidade dos textos, foram realizados experimentos de pré-processamento para a identificação das melhores técnicas a serem utilizadas para esse tipo de conteúdo. Em comparação com a baseline do trabalho, foi possível alcançar um novo patamar de resultados, excedendo 90% de acurácia em grande parte dos modelos treinados. Destacam-se os modelos Longformer + CNN que atingiu 94% de acurácia com 100% de precisão e o modelo Word2Vec + CNN com 93,55% de acurácia. As descobertas do estudo representam uma aplicação bem-sucedida de inteligência artificial na administração pública.

**Palavras-chave**: aprendizado profundo, classificação de textos longos, transferência de aprendizado, poucos dados, CNN, LSTM, DNN, *word embeddings*, *Longformer*, *Word2Vec*, PPF, MCTI.

# Abstract

Natural language processing (NLP) is an area of artificial intelligence that has been gaining much attention in recent years. The great recent advances attracted the attention of the Ministry of Science, Technology, and Innovations (MCTI) to the execution of a project to locate international funding for research and development accessible to Brazilian researchers through its Research Financing Products Portfolio (FPP). Classification presents a challenge for this solution due to the absence of high-quality labeled data, a requirement for most state-of-the-art implementations in the field. This work explores different machine learning strategies to classify long, unstructured, and irregular texts obtained by scraping websites of funding institutions to, through an incremental approach, find a suitable method with good performance. Due to the limited data available for supervised training, pre-training solutions were employed to learn the context of words from other datasets with significant similarity and larger sizes. Then, using the acquired information, a transfer of learning associated with deep learning models was applied to improve the understanding of each sentence. In order to reduce the impact of text irregularity, pre-processing experiments were carried out to identify the best techniques for this type of content. Compared to the baseline of the work, it was possible to reach a new level of results, exceeding 90% accuracy in most of the trained models. The Longformer + CNN model reached 94% accuracy with 100% precision, and the Word2Vec + CNN model achieved 93.55% accuracy. The study's findings represent a successful application of artificial intelligence in public administration.

**Keywords**: deep learning, long text classification, transfer learning, limited size datasets, CNN, LSTM, DNN, word embeddings, *Longformer*, *Word2Vec*, PPF, MCTI.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| ACT | Attentive Convolutional Transformer . . . . . 34 |
| BERT | Bidirectional Encoder Representations from Transformers . . . . . 33 |
| BOW | Bag of Words . . . . . 29 |
| CD | Ciência de Dados . . . . . 29 |
| CNN | Convolutional Neural Network . . . . . 32 |
| CT&I | Ciência, Tecnologia e Inovações . . . . . 21 |
| DNN | Deep Neural Network . . . . . 32 |
| GCNN | Graph Convolutional Neural Networks . . . . . 32 |
| GNN | Graph Neural Network . . . . . 32 |
| GPT | Generative Pre-trained Transformer . . . . . 33 |
| GT | Grupo de Trabalho . . . . . 29 |
| HTML | Hypertext Markup Language - Linguagem de escrita web . . . . . 62 |
| IA | Inteligência Artificial . . . . . 21 |
| LSTM | Long Short-term Memory . . . . . 32 |
| MCTI | Ministério da Ciência, Tecnologia e Inovações . . . . . 21 |
| MLP | Multilayer Perceptron . . . . . 47 |
| NB | Naive Bayes . . . . . 29 |
| PLN | Processamento de Linguagem Natural . . . . . 21 |
| PPF | Portfólio de Produtos Financeiros . . . . . 21 |
| RF | Random Forest . . . . . 29 |
| RNA | Redes Neurais Artificiais . . . . . 33 |
| RNN | Recurrent Neural Network . . . . . 32 |
| SEFIP | Secretaria de Estruturas Financeiras e de Projetos . . . . . 22 |
| SVM | Support Vector Machine . . . . . 29 |
| TF-IDF | Term Frequency-inverse Document Frequency . . . . . 29 |

# Sumário

# 1 Introdução

A Inteligência Artificial (IA) é uma das principais tecnologias disruptivas da atualidade e seu uso tem mudado a forma como processos organizacionais, operacionais e logísticos têm sido feitos em grandes corporações e órgãos governamentais. O amadurecimento e expansão da IA e ciência de dados têm possibilitado autonomia e eficiência às máquinas para resolução de problemas complexos em diversas áreas e setores de aplicações. Alguns dos benefícios do uso de IA são: automação do processo de análise de dados, melhoria na tomada de decisões, redução de erros e custos operacionais.

Dentro desse contexto, o Processamento de Linguagem Natural (PLN) surge como a vertente de IA que ajuda os computadores a entender, interpretar e manipular a linguagem humana, permitindo que eles desempenhem diversas tarefas envolvendo informações textuais. Um problema tradicional dentro de PLN é a classificação, que consiste em prever ou atribuir uma categoria predefinida a uma entrada de texto.

## 1.1 Motivação

Os avanços recentes na área de Inteligência Artificial despertaram o interesse do Ministério da Ciência, Tecnologia e Inovações (MCTI) em automatizar o processo de identificação e seleção de oportunidades relevantes para projetos de pesquisa brasileiros através do Portfólio de Produtos Financeiros (PPF). O PPF visa organizar e disponibilizar informações sobre fontes de financiamento de projetos de ciência, tecnologia e inovação (CT&I) e oportunidades para os pesquisadores brasileiros (SILVA, B. et al., 2021). Essa iniciativa desempenha um papel fundamental na promoção da ciência e tecnologia pelo país.

Essa pesquisa consiste na utilização de algoritmos de Processamento de Linguagem Natural (PLN) para a classificação dessas oportunidades quanto à elegibilidade para projetos brasileiros. Ele está inserido dentro de um projeto financiado pelo MCTI que abrange, além da classificação das oportunidades, a busca automatizada em plataformas digitais conhecidas, através de técnicas de raspagem de dados. Outros focos do projeto relacionado consistem na síntese dos textos das oportunidades para facilitação de sua divulgação e a implementação de um protótipo de sistema que utiliza aprendizagem por reforço e a experiência dos usuários para melhoria dos resultados obtidos.

O processo de classificação se destaca especialmente por servir como um portão para definir as oportunidades úteis de serem divulgadas e que passarão para a etapa de síntese. Dentro das expectativas do Ministério com o projeto, a meta de classificação é a mais importante porque consolida o resultado do projeto em questão de automação do trabalho

e utilidade da ferramenta entregue, uma vez que, com uma performance insatisfatória na classificação, a ferramenta perde a maior parte de seu valor.

Também é válido destacar a importância intrínseca do projeto no contexto de fomento à pesquisa e ao estreitamento das relações da Universidade de Brasília com o Ministério, o que é algo valiosíssimo. Além disso, a parceria demonstra a vontade do ministério de apoiar a pesquisa, e a capacidade da universidade de entregar resultados competitivos com o mercado, mesmo com recursos limitados em comparação com as grandes empresas da área.

## 1.2 Definição do Problema

Este projeto irá focar no problema específico da classificação do PPF. A descrição do problema pode ser vista na Figura 1.1.

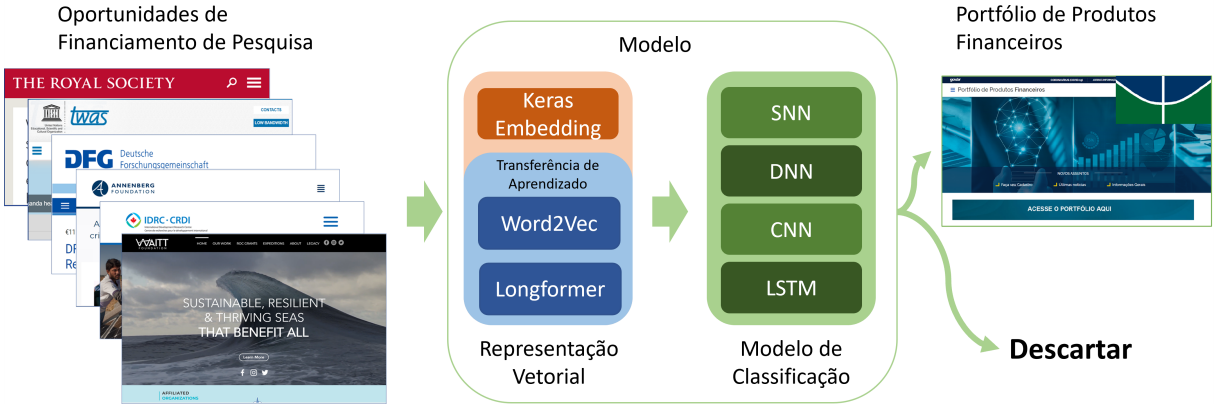



Figura 1.1 – Esquemático do problema. Traduzida de (ROCHA. et al., 2022)

Os dados de entrada consistem em textos contendo descrições de oportunidades de financiamento oferecidos por diversas instituições ao redor do mundo nos seus websites como bolsas de estudo, subsídios e outros. Uma pequena parte desses dados foi manualmente catalogada pela equipe do MCTI e foi utilizada para o treinamento supervisionado do modelo de classificação que possui sua acurácia definida pela discriminação das oportunidades que permitem financiamento para projetos brasileiros.

Um grande desafio previsto diz respeito à qualidade dos dados obtidos através da raspagem. Devido às diferenças no formato, na linguagem e na escrita do texto de cada uma das plataformas raspadas, não existe padronização dos dados de entrada, o que pode ocasionar uma queda de performance ou uma piora nos resultados da etapa de classificação.

Outro desafio é o tamanho dos textos, que pode chegar a mais de trinta mil caracteres e tornar o trabalho da classificação bastante custoso computacionalmente.

Ainda mais um desafio é a quantidade pequena de dados catalogados para o treinamento supervisionado. Enquanto modelos estado da arte em classificação utilizam datasets

contendo milhares de dados para treino, o projeto encontra-se limitado a uma pequena quantidade de dados catalogada manualmente.

Como será comentado no capítulo 3, houve tentativas de classificar um conjunto similar de dados utilizando técnicas de aprendizado de máquina e PNL. Neste trabalho, será implementado um sistema que faz uso de técnicas de Aprendizado Profundo e estratégias de treinamento estado-da-arte para superar os resultados obtidos anteriormente.

A saída do sistema deverá ser uma nova planilha contendo apenas as oportunidades que são consideradas elegíveis, e servirá de entrada para as próximas etapas do projeto.

## 1.3 Objetivos

### 1.3.1 Objetivo Geral

O objetivo principal consiste em desenvolver métodos de processamento de linguagem natural e aprendizagem profunda para classificar um conjunto pequeno de textos longos e não estruturados relativos a oportunidades adquiridas através de buscas automatizadas em plataformas digitais conhecidas. A classificação em termos de elegibilidade para projetos brasileiros permitirá selecionar as oportunidades para que possam integrar o portfólio de produtos financeiros organizado pelo MCTI.

### 1.3.2 Objetivos Específicos

Para atingir o objetivo geral deste trabalho foram estabelecidos os objetivos específicos a seguir:

- Estudo do estado da arte e técnicas de aprendizagem de máquina para modelar estratégia de definição dos experimentos com intenção de encontrar o melhor modelo de classificação;
- Análise da qualidade dos dados obtidos na raspagem. Apresentar os dados coletados, incluindo os textos de oportunidades financeiras fornecidos por várias instituições de apoio à pesquisa ao redor do mundo;
- Introdução de uma estratégia de pré processamento dos dados para simplificação da representação e melhoria da eficiência destes no modelo;
- Utilização de uma estratégia de “pré-aprendizagem” para aprender as features a partir de dados não-estruturados em larga escala seguida de métodos de *transfer learning* para melhorar a performance do sistema;

- Realização de experimentos com métricas de avaliação bem definidas para encontrar o melhor modelo de classificação dada às restrições de infraestrutura. Utilizando técnicas de transferência de aprendizado e representação vetorial de dados;
- Implementação de um modelo final de classificação com resultados satisfatórios aprovados pelo MCTI e disponibilização de um notebook e uma aplicação prontas para uso.

## 1.4 Aspectos Metodológicos

Levando em consideração os recursos computacionais e o esforço de implementação necessários para a obtenção de bons resultados a metodologia proposta é a de realizar abordagens incrementais, realizando testes com arquiteturas e estratégias mais simples e, após a análise dos resultados obtidos em cada etapa, sugerir melhorias de acordo com o diagnóstico considerado. A metodologia usada no desenvolvimento do presente trabalho é composta das seguintes etapas:

- Pesquisa de revisão bibliográfica do estado da arte sobre aprendizado de máquina, redes neurais artificiais profundas, transferência de aprendizado e *transformers*;
- Modelagem e Desenvolvimento de arquiteturas de redes neurais baseadas no estado da arte para realizar a classificação das oportunidades. Deseja-se comparar diferentes modelos para encontrar o de melhor performance;
- Treinamento não-supervisionado dos modelos para realização de transferência de aprendizado;
- Treinamento supervisionado dos modelos para a determinação de acurácia obtida;
- Análise dos resultados obtidos de maneira a observar e apontar melhorias possíveis para o modelo;
- Implementação de melhorias no modelo, no que diz respeito ao tratamento dos dados e qualidade da entrada para a rede;
- Experimentação de técnicas de pré-processamento para obter os melhores resultados possíveis;
- Novos treinamentos utilizando os dados pré-processados e os melhores modelos de aprendizagem de máquina;
- Avaliação dos resultados e definição do melhor modelo a ser entregue para o ministério.

## 1.5 Contribuição do Trabalho

Este trabalho tem como contribuição científica o artigo *"Using transfer learning to classify long unstructured texts with small amounts of labeled data"* apresentado no congresso WEBIST 2022, que pode ser lido no Anexo A.

## 1.6 Organização do Documento

Este trabalho está dividido em 9 capítulos. O capítulo 2 apresenta a descrição do projeto correlato que motivou o desenvolvimento do presente trabalho, e retrata também as problemáticas, desafios envolvidos e soluções elencadas a serem exploradas.

O Capítulo 3 contempla a revisão bibliográfica acerca da tarefa de classificação em processamento de linguagem natural através do estudo do trabalho precursor, bem como uma revisão sistemática semi automatizada para elencar o estado da arte e trabalhos relacionados ao tema.

O Capítulo 4 aborda a fundamentação teórica relativa às temáticas de aprendizagem de máquina, processamento de linguagem natural e classificação, necessárias para a contextualização e desenvolvimento do presente projeto.

O Capítulo 5 contempla a descrição dos *datasets* utilizados para o treinamento dos modelos utilizados no trabalho, apresentando exemplos dos textos utilizados e comentando sobre a sua estrutura.

O Capítulo 6 apresenta a metodologia e implementações desenvolvidas para resolução da problemática central do projeto, explicitando o processo de desenvolvimento, a modelagem das soluções implementadas, assim como as descobertas e dificuldades encontradas.

O Capítulo 7 aborda os resultados dos experimentos realizados, trazendo uma análise detalhada de cada abordagem desenvolvida.

O Capítulo 8 contempla as discussões e considerações finais, comentando os resultados obtidos considerando a metodologia utilizada e apresentando a abordagem que melhor se adequa à problemática inicial.

O Capítulo 9 apresenta a conclusão do trabalho e sugestões para trabalhos futuros.

# 2 Projeto Relacionado

O Portfólio de Produtos Financeiros (PPF) é um instrumento criado e mantido pela Secretaria de Estruturas Financeiras e de Projetos (SEFIP) e pelo Ministério de Ciência, Tecnologia e Inovações (MCTI), cujo objetivo é auxiliar o fomento de pesquisas científicas e a captação de recursos não orçamentários e financiamentos. O PPF apresenta informações sobre oportunidades ofertadas por fundações, bancos e instituições nacionais e internacionais, como Agências de Fomento e Banco Multilaterais, dando assim, maior visibilidade às oportunidades existentes e ao desenvolvimento científico e tecnológico do país.

Atualmente, no entanto, as ferramentas de buscas e atualização de informações do sistema PPF são feitas manualmente, o que torna o processo longo, lento e suscetível a erros. A expectativa é que a modernização do sistema proporcione ferramentas de buscas e atualizações de dados de maneira automática e otimizada.

O projeto visa, utilizando um enfoque ágil, identificar e implementar buscas automatizadas em endereços eletrônicos conhecidos das instituições provedoras de oportunidades científicas, para que seja possível a realização de um processo de análise e categorização das ofertas. Deseja-se também implementar mecanismos de disponibilização da solução com recomendação e interação com o usuário para que sua experiência permita o aperfeiçoamento contínuo da ferramenta e a usabilidade do recurso, fazendo uso de aprendizagem por reforço.

Além da automação do processo atual de identificação e seleção de oportunidades, que é o escopo central da presente pesquisa, destaca-se também que a tecnologia sendo desenvolvida possui possíveis aplicações em diferentes áreas da administração pública.

Os benefícios do processo de automação integrado ao serviço público irá refletir na melhoria do atendimento ao cidadão e na entrega oportuna dos serviços. Além disso, a realocação do tempo dos servidores das tarefas operacionais manuais permitirá um foco e dedicação maiores em tarefas mais estratégicas.

Apesar do foco deste trabalho ser a classificação, este capítulo descreve brevemente todas as etapas do projeto: Raspagem de dados, Classificação, Síntese e Recomendação. Além disso, neste capítulo também é apresentado o cronograma do projeto, assim como as metas relacionadas a ele.

## 2.1 Raspagem

A primeira etapa na automação do processo de identificação e avaliação dessas fontes é a leitura ou captura dos dados. Para isso, foi feito uso de uma técnica chamada de Raspagem

de dados *(web scraping)* que consiste no acesso automatizado às principais plataformas online que são fontes de recursos financeiros para projetos de pesquisa. A ideia foi conseguir varrer os dados desses *websites* e coletar as informações referentes às oportunidades, para que elas possam ser utilizadas nas próximas etapas do projeto.

Devido ao interesse do projeto ser a automatização de um processo já existente, existia a necessidade da implementação de *scripts* em *python* para a realização da raspagem em cima de plataformas já mapeadas, que possuem uma estrutura definida e onde os dados são atualizados dinamicamente. Essa lista específica de plataformas limita a quantidade de dados disponível para o aprendizado supervisionado realizado nas etapas seguintes, uma vez que não se deseja buscar na internet quaisquer outras oportunidades para incrementar a quantidade de dados.

O foco da raspagem consiste no conteúdo textual principal da página sobre as informações de oportunidades de pesquisa, desconsiderando informações secundárias como URL, título, cabeçalho, entre outros metadados. Um detalhe importante é que existe a necessidade da criação de *scripts* diferentes para cada plataforma já que as estruturas dos sites diferem entre si. Outro detalhe a se evidenciar é que as alterações nessas estruturas podem tornar os *scripts* de raspagem obsoletos e poderão necessitar de manutenções periódicas.

A entrega desta etapa são planilhas contendo as informações textuais relativas às oportunidades encontradas em cada plataforma, que são posteriormente consolidadas em um dataset contendo as informações relativas a todas as plataformas raspadas.

## 2.2 Classificação

Com os dados devidamente coletados e organizados podemos fazer uso de técnicas de aprendizagem de máquina *(machine learning)* para realizar a classificação das oportunidades, de maneira a selecionar aquelas que possuem elegibilidade para projetos brasileiros. Como a classificação será feita com dados em formato de texto, será necessário utilizar técnicas de processamento de linguagem natural (PLN) para que seja possível extrair informações contextuais e permitir a interpretação do texto pelo computador.

O trabalho (SILVA, B. et al., 2021) realizado anteriormente fez uso de algumas técnicas de aprendizado de máquina para classificar esse tipo de oportunidade e, aqui deseja-se aprofundar no assunto através da utilização de aprendizado profundo. Para contornar as limitações causadas pela pequena quantidade de dados catalogados para o treinamento da rede, decidiu-se fazer uso de técnicas de transferência de aprendizado de maneira a melhorar o desempenho da classificação.

No final desta etapa o entregável é um modelo de classificação validado capaz de tratar as planilhas contendo dados de oportunidades e filtrar apenas aquelas que são elegíveis

para projetos brasileiros.

## 2.3 Síntese

A etapa seguinte é a de síntese dos dados que consiste na utilização de técnicas de aprendizagem de máquina e processamento de linguagem natural para realizar a sumarização ou resumo dos textos das oportunidades selecionadas e classificadas. Os textos sintetizados permitirão maior facilidade na divulgação das oportunidades.

Nesta etapa os textos das oportunidades já classificadas como elegíveis serão introduzidos no modelo de sumarização que farão o processamento do texto de maneira a reduzir o seu tamanho preservando as informações relevantes. Será feito uso de métodos de extração de informações relevantes do texto e *Transformers* (Seção 4.5) para que seja possível realizar a compreensão do texto e ser capaz de resumi-lo. Essas informações condensadas irão auxiliar a disponibilidade e transparência para pesquisadores brasileiros.

## 2.4 Recomendação pelo Aprendizado de Reforço

De maneira a permitir a melhoria contínua da solução proposta, a última etapa do projeto envolve o desenvolvimento de um protótipo inicial de um sistema de recomendações que faça uso de aprendizagem de máquina. A implementação do mecanismo de recomendação utilizará de aprendizado por reforço, técnicas de IA e um feedback a respeito da experiência dos usuários.

Para isso, será traçado um perfil do usuário para que seja possível recomendar as oportunidades que fazem sentido para área e contexto de seu perfil. De acordo com a ajuda do usuário, esse perfil vai se adaptando e o sistema conseguirá recomendar oportunidades de maneira mais precisa.

Nesta etapa deseja-se especificar e modelar esse sistema, porém o projeto se limita a entregar uma plataforma que possa ser desenvolvida futuramente de maneira a ser acoplada na atual experiência do *website* do PPF[1].

## 2.5 Metas e Cronograma do Projeto

A Tabela 2.1 apresenta as metas definidas e apresentadas como parte da estrutura analítica do projeto relacionado. Nela estão destacadas as metas que pertencem à equipe de classificação e que serão o foco deste trabalho.

[1] Disponível em: https://ppf.mcti.gov.br/

Tabela 2.1 – Metas e cronograma do projeto relacionado, com destaque nos itens referentes a este trabalho.

| **Meta** | **Atividades** | **Produto** | **Duração** | | |
|---|---|---|---|---|---|
| | | | Início | Término | Duração (dias) |
| 1 | Estrutura Analítica do Projeto, mobilização da equipe, metodologias e ferramental, contratação de fundação de apoio | EAP | 10/01/2022 | 15/02/2022 | 36 |
| 2 | Especificação detalhada: análise das atividades, alinhamento dos problemas, divisão de tarefas, elaboração de indicadores e levantamento de dados internos | Relatório técnico de especificação detalhada do Projeto de Pesquisa | 17/02/2022 | 18/03/2022 | 29 |
| 3 | Raspagem de dados em sites e em artigos com informações sobre fontes de financiamento | Produto: GitHub com *scripts* para raspagem de dados em API de artigos e sites de fontes de financiamento | 31/01/2022 | 29/04/2022 | 88 |
| 4 | Identificação de oportunidades de financiamento elegíveis para o portfólio de produtos financeiros | Produto: Banco de dados de oportunidades de financiamento, com rótulo de elegibilidade para portfólio de produtos financeiros | 07/03/2022 | 27/05/2022 | 81 |
| 5 | Metodologia semi-automatizada para Revisão Sistemática da Literatura | Minuta de artigo científico | 14/02/2022 | 06/05/2022 | 81 |

| | | | | | |
|---|---|---|---|---|---|
| 6 | Revisão da Literatura sobre Processamento de Linguagem Natural para classificação (elegibilidade) | Minuta de artigo científico | 31/01/2022 | 19/05/2022 | 108 |
| 7 | Revisão da Literatura sobre Processamento de Linguagem Natural para síntese (geração de texto) | Minuta de artigo científico | 11/07/2022 | 12/09/2022 | 63 |
| 8 | Revisão da Literatura sobre Sistemas de Recomendação com Aprendizado sob Reforço | Minuta de artigo científico | 30/05/2022 | 29/07/2022 | 60 |
| 9 | Pré-processamento de dados dos sites e artigos, para classificação de oportunidades de financiamento elegíveis para o portfólio de produtos financeiros | Produto: GitHub com *scripts* para préprocessamento de dados de artigos e sites de fontes de financiamento | 20/05/2022 | 15/07/2022 | 56 |
| 10 | Aprendizado de máquina com processamento de linguagem natural (PLN) e classificação de oportunidades de financiamento | Produto: GitHub com *scripts* de PLN e classificação de dados de artigos e sites de fontes de financiamento | 19/05/2022 | 19/08/2022 | 92 |
| 11 | Aprendizado de máquina com processamento de linguagem natural (PLN) para síntese e geração de textos de oportunidades de financiamento | Produto: GitHub com *scripts* de PLN para síntese e geração de texto a partir de dados de artigos e sites de fontes de financiamento | 12/09/2022 | 18/11/2022 | 67 |

| | | | | | |
|---|---|---|---|---|---|
| 12 | Aprendizado de máquina para recomendação de oportunidades de financiamento, a partir de feedback dos usuários | Produto: GitHub com *scripts* de recomendação e aprendizado sob reforço, a partir de dados de artigos e sites de fontes de financiamento e feedback dos usuários | 01/08/2022 | 16/10/2022 | 76 |
| 13 | Testes e validação dos *scripts* | Produto: GitHub com *scripts* de teste e validação | 22/08/2022 | 23/12/2022 | 123 |
| 14 | Especificação de protótipo online da versão automatizada do portfólio de produtos financeiros | Produto: Documento de especificação sobre plataforma para disponibilização de dados e levantamento de feedback para treinamento | 17/10/2022 | 14/01/2023 | 89 |
| 15 | Elaboração de versões finais do sistema e da documentação | Produto: GitHub final do projeto e relatórios finais da automação do portfólio de produtos financeiros | 17/10/2022 | 14/01/2023 | 89 |

A meta 6 diz respeito à revisão sistemática de literatura necessária para dar o embasamento teórico para a tarefa de classificação dentro do contexto do projeto. Essa revisão é brevemente descrita na Seção 3.2, e o resultado entregue, além de um relatório descrevendo os achados, foi a submissão do artigo (ROCHA. et al., 2022) apresentado no congresso WEBIST, onde foram destacadas as técnicas mais adequadas para o problema aplicado a um dataset preliminar, uma vez que o dataset final do projeto só ficou disponível após a entrega da Meta 4.

A meta 9 aborda o pré-processamento necessário para otimizar o processo de classificação. Essa etapa é descrita na Seção 6.4 e os resultados obtidos estão apresentados na Seção 7.2.

A meta 10 aborda a classificação a ser realizada acima do dataset final entregue pela Meta 4 já pré-processado pela entrega da Meta 9. Essa etapa é descrita na Seção 6.5 e seus resultados estão apresentados na Seção 7.3.

As metas 13, 14 e 15 dizem respeito a validação dos modelos de classificação, síntese e recomendação, seguido pelas suas respectivas implantações com a entrega de uma ferramenta pronta para o uso do MCTI. No contexto da classificação, essas metas são brevemente descritas na Seção 6.6 e Capítulo 8.

# 3 Revisão Bibliográfica

Esta etapa investigou na literatura científica o estado da arte em processos de Classificação em Processamento de Linguagem Natural (PLN) para automatização do sistema PPF. A busca se iniciou no trabalho de base realizado anteriormente por (SILVA, B. et al., 2021), onde foram desenvolvidos experimentos e abordagens para a classificação em nível de elegibilidade para uma base de dados de oportunidade de financiamento, similar à base de dados utilizada no presente projeto.

Além do trabalho anterior, foi utilizado o *Semantic Scholar* como base de pesquisa de artigos científicos para a revisão sistemática do estado da arte. O *Semantic* consiste em uma plataforma de busca semi automatizada de publicações acadêmicas lançada em novembro de 2015 e desenvolvida pelo *Allen Institute for Artificial Intelligence*, e conta com mecanismos de buscas regidos por inteligência artificial.

## 3.1 Trabalho Anterior

Este problema foi estudado pelo Grupo de Trabalho do MCTI sob a óptica da ciência de dados, com uso de técnicas de Varredura de Dados, Aprendizado de Máquina, Inteligência Artificial, Processamento de Linguagem Natural e as intersecções entre essas áreas, utilizando *Bag of Words* (BOW), *Term Frequency-inverse Document Frequency* (TF-IDF), *Naive Bayes* (NB), *Support Vector Machine* (SVM) e *Random Forest* (RF) (SILVA, B. et al., 2021).

### 3.1.1 Baseline do Trabalho

Em conformidade com as estratégias de transformação digital do governo, a SEFIP organizou um Grupo de Trabalho (GT) com o intuito de desenvolver um projeto para automatizar o processo de busca, classificação e recomendação de oportunidades de financiamento e pesquisas científicas do PPF, utilizando para tal, Ciência de Dados (CD) e Inteligência Artificial (IA). (BRASIL, 2019)

No referido projeto, para a realização da tarefa de classificação, foram utilizadas as metodologias de *Naive Bayes, Support Vector Machines* e *Random Forest*, associadas às técnicas de representação *Bag of Words* (BOW) e *Term Frequency-Inverse Document Frequency* (TF-IDF). Para avaliar a classificação, foi realizada uma análise cruzada entre as três metodologias e as duas técnicas de representação vetorial, e, para cada combinação, foram selecionados e apresentados os melhores resultados de acurácia. (SILVA, B. et al., 2021).

Os resultados obtidos pelo trabalho podem ser observados na Tabela 3.1.

Tabela 3.1 – Resultados obtidos. Fonte: (SILVA, B. et al., 2021)

| | **Acurácia** | **F1-Score** | **Precision** | **Recall** |
|---|---|---|---|---|
| NB + BOW | 0,82 | 0,81 | 0,81 | 0,82 |
| NB + TF-IDF | 0,86 | 0,85 | 0,85 | 0,85 |
| SVM + BOW | 0,78 | 0,76 | 0,76 | 0,76 |
| SVM + TF-IDF | 0,82 | 0,80 | 0,81 | 0,79 |
| RF + BOW | 0,84 | 0,82 | 0,85 | 0,80 |
| RF + TF-IDF | 0,82 | 0,80 | 0,81 | 0,79 |

Os resultados foram positivos e promissores dentro do contexto de avaliação e classificação de oportunidades de financiamento, principalmente considerando que o projeto contava com apenas 200 dados para treinamento. O melhor resultado foi obtido pela metodologia de *Naive Bayes*, que apresentou bom desempenho para pequenos dados amostrais, tendo alcançado uma acurácia de 86%. O trabalho ainda destaca como propostas de trabalhos futuros, a utilização de aprendizado profundo no processo de classificação, objeto de pesquisa do presente projeto em desenvolvimento.

## 3.2 Revisão Sistemática

Conforme comentado anteriormente, foi realizada uma busca semi automatizada através da plataforma *Semantic Scholar* (AI2, 2023), que consiste num mecanismo de busca de pesquisas acadêmicas utilizando inteligência artificial. O mecanismo de automação utilizado foi o algoritmo produzido na Meta 5 do projeto relacionado, cuja teoria está demonstrada no artigo *"Few-shot approach for systematic literature review classification"* (MELO et al., 2022). Este mecanismo utiliza o título e o resumo de alguns exemplos rotulados à priori e ranqueia a base de dados em ordem de confiança para incluir na revisão sistemática. Este processo é eficiente para acelerar o processo de revisão dos artigos por eliminar os artigos que não possuem evidência suficiente para serem incluídos na revisão sistemática.

A busca fez uso de palavras chaves consideradas relevantes para o contexto do projeto, utilizando as projeções futuras do trabalho (SILVA, B. et al., 2021) e a perspectiva inicial da necessidade de utilização de transferência de aprendizado para contornar a pequena quantidade de dados rotulados para o treinamento das redes. A estrutura utilizada na pesquisa é apresentada na Figura 3.1:

A revisão sistemática de literatura utilizou a estrutura de fases e etapas definidas por (KITCHENHAM; CHARTERS, 2007), considerada a base para revisões sistemáticas na engenharia de *software*, e resultou em um relatório completo explicitando todos os critérios utilizados de busca e análise [1]. No total foram 99 artigos selecionados para a leitura, abordando a utilização de: Redes Neurais Recorrentes (RNN em inglês), Redes Neurais Profundas

[1] Relatório está disponível no github do projeto: `https://github.com/chap0lin/PPF-MCTI`

**PALAVRAS CHAVE UTILIZADAS NA PESQUISA SISTEMÁTICA**

("text classification")
**AND**
("vectorization" **OR** "vector representation" **OR**
"document representation" **OR** "word representation" **OR**
"keras" **OR** "word2vec" **OR** "doc2vec" **OR**
"long text classification" **OR** "long document classification" **OR**
"longformer" **OR** "document-level" **OR** "sentence-level" **OR**
"transfer learning" **OR** "pre-training")
**AND**
("DNN" **OR** "deep neural network" **OR**
"CNN" **OR** "convolutional neural network" **OR** "convolutional network" **OR**
"LSTM" **OR** "long-short term memory")

Figura 3.1 – Palavras chave utilizadas para realizar a pesquisa sistemática.

(DNN em inglês), Redes Neurais Convolucionais (CNN em inglês), Redes Neurais de Longa memória de curto prazo (LSTM em inglês), *Transformers*, entre outras. As Tabelas 3.2 e 3.3 apresentam distribuições dos artigos por técnica utilizada e por modelo utilizado, respectivamente, e nelas é possível observar algumas tendências de processamento de linguagem natural.

Tabela 3.2 – Distribuição dos artigos por técnica utilizada.

| **Técnica Principal** | **Número de Artigos** | **%** |
|---|---|---|
| Attention | 22 | 22% |
| Embedding | 20 | 20% |
| Features | 7 | 7% |
| Pre-Training | 5 | 5% |
| Transfer Learning | 5 | 5% |
| Encoder | 4 | 4% |
| BoW | 4 | 4% |
| BERT-Based | 3 | 3% |
| GloVe | 3 | 3% |
| Word2vec | 2 | 2% |
| TF-IDF | 2 | 2% |
| Outros | 22 | 22% |
| **Total de Artigos** | **99** | **100%** |

Tabela 3.3 – Distribuição dos artigos por modelo utilizado.

| **Modelo Principal** | **Número de Artigos** | **%** |
|---|---|---|
| CNN | 31 | 31% |
| DNN | 16 | 16% |
| LSTM | 12 | 12% |
| Transformers | 7 | 7% |
| Adversarial Attack | 6 | 6% |
| GCNN | 6 | 6% |
| GNN | 4 | 4% |
| RNN | 4 | 4% |
| Capsule Network | 3 | 3% |
| Outros | 10 | 10% |
| **Total de Artigos** | **99** | **100%** |

Neste capítulo será apresentado um breve resumo dos resultados obtidos, considerando os objetivos da pesquisa.

### 3.2.1 Classificação em NLP

No contexto de PLN, a classificação consiste em prever ou atribuir uma categoria predefinida a uma entrada de texto. Para isso, uma etapa fundamental é a representação do texto. Na literatura podem ser encontrados diversos modelos de redes neurais que realizam aprendizado a partir de representações textuais desde modelos convolucionais (ZHANG, L. et al., 2017); (SHEN et al., 2018) e recorrentes (YOGATAMA et al., 2017); (SEO et al., 2017) até mecanismos de atenção (YANG, Z. et al., 2016); (LIN, Z. et al., 2017).

Outra alternativa é a utilização de modelos pré-treinados em grandes *Corpus* de dados, que tem alcançado excelente performance para classificação e outras tarefas em PLN e, possivelmente, sem a necessidade de treinar o novo modelo do zero. Um tipo de modelo é o de *Word Embeddings* como *Word2Vec* (MIKOLOV; SUTSKEVER et al., 2013) e *GloVe* (PENNINGTON et al., 2014). Outra opção é o uso de *Word Embeddings* contextualizados como *CoVe* (MCCANN et al., 2017) and *ELMo* (PETERS et al., 2018).

Muitas das vezes esses *Word Embeddings*, contextualizados ou não, são utilizados como características adicionais para auxiliar a tarefa principal. Estudos mais recentes têm mostrado que modelos pré-treinados são efetivos em aprender representações comuns de linguagem ao utilizar altas quantidades de dados não catalogados, como o GPT (do inglês

*Generative Pre-trained Transformer* ou *Transformer* Generativo Pré-treinado) (RADFORD et al., 2018); (BROWN et al., 2020) e BERT (do inglês *Bidirectional Encoder Representations from Transformers* ou Representações de Encoder Bidimensionais de *Transformers*) (DEVLIN et al., 2018).

A maior parte desses modelos passa pelo processo de pré-treino, utilizando quantidades finitas de dados, e os resultados excepcionais dependem do volume dos dados catalogados e de alta qualidade. A possível variação na quantidade ou qualidade desses dados pode influenciar nos resultados. Em aplicações reais, esses modelos podem apresentar resultados inesperados ou insatisfatórios afetando a robustez da solução (GRON, 2017).

No geral existem diversos modelos de alto desempenho que são utilizados para classificação, porém a literatura não apresenta uma resposta definitiva de qual modelo que melhor se adequa às necessidades do projeto. Nesta seção serão apresentados alguns modelos que serviram como suporte para a arquitetura do modelo desenvolvido no projeto.

#### 3.2.1.1 Aprendizagem Profunda

Os avanços recentes no poder computacional e na disponibilidade de dados permitiram à Aprendizagem Profunda uma ressurgência (THOMPSON et al., 2020), mais especificamente, o progresso de Redes Neurais Artificiais (RNA) como Redes Neurais Convolucionais (no inglês CNN ou *Convolutional Neural Networks*) e LSTM (do inglês *Long Short-term Memory*). Esses modelos de aprendizagem profunda têm sido bem sucedidos em classificação de textos e documentos (MINAEE et al., 2021).

(ZHOU, 2022) utiliza, em modelos de LSTM e CNN, *Term Frequency-inverse Document Frequency* (TF-IDF) para remover características com pesos menores, extrair características chave no texto, extrair o vetor de palavras correspondente por meio de *Word2Vec* e, em seguida, inseri-lo no modelo CNN-LSTM. (ZHAO et al., 2021) desenvolveu uma abordagem que utiliza DNNs para produzir modelos auto interpretáveis de classificação de texto com desempenho comparável a modelos CNNs mais complexos. (LI, J. et al., 2022), para superar a falha do *Word2Vec* em capturar totalmente a informação semântica do texto, propõe um *Transformer* hierárquico modelo CNN e aplica na classificação multirrótulo. Um modelo *Transformer-CNN* é construído para capturar informações semânticas de diferentes níveis do texto (nível de palavras e frases), usando mecanismo de auto-atenção multidirecionada (*multi-headed self-attention*), e uma rede neural convolucional para extrair os principais recursos semânticos.

(WANG, J. et al., 2017) e (XIAO et al., 2018) fizeram uso de redes LSTM associadas com *Word Embeddings* para alcançar resultados excelentes em classificação de texto. (SEMBERECKI; MACIEJEWSKI, 2017) aplicaram redes neurais profundas (DNN) com LSTM, realizando testes com diferentes abordagens para a representação textual, e destacaram os resultados do *Word2Vec* em relação à técnicas de *Bag of Words* (BOW). (KIM, 2014) realizou

uma série de experimentos utilizando redes neurais convolucionais (CNNs) treinadas a partir de representações vetoriais pré-treinadas para tarefas de classificação e alcançou resultados excelentes.

(LI, P. et al., 2021) apresentam um *Attentive Convolutional Transformer* (ACT) que aproveita as vantagens do Transformer e da CNN para classificação de texto. Propõem novo mecanismo de convolução atenta, que utiliza significado semântico dos filtros convolucionais atentamente para transformar o texto de um espaço de palavras complexo em um espaço de filtro convolucional mais informativo, onde $n$-grams importantes são capturados. Experimentos em várias tarefas de classificação de texto e análises detalhadas mostram que o ACT é um classificador de texto universal leve, rápido e eficaz, superando CNNs, RNNs e modelos atenciosos, incluindo o Transformer.

(WANG, Z.; QU, 2017) propõe um novo método de classificação de texto extraído da Web, baseado em CNN e SVM aprimorados, usando o modelo CNN com estrutura de cinco camadas para extrair características do texto, classificar e prever usando SVM.

(ALACHRAM et al., 2020) usaram a abordagem *Word2Vec* para gerar representações de vetores de palavras com base em um corpus composto por mais de 16 milhões de resumos *PubMed*. Um *pipeline* de mineração de texto para produzir *Word2Vec Embeddings* com diferentes propriedades foi experimentado para avaliar sua utilidade para análise biomédica. Comparações com bancos de dados biológicos mostraram que as representações de palavras produzidas por algoritmos de mineração de texto como *Word2Vec* foram capazes de capturar relações biologicamente significativas entre entidades.

### 3.2.2 Textos Longos

Apesar das abordagens que utilizam *Transformers* terem alcançado o estado da arte em tarefas de processamento de linguagem natural, elas são mais adequadas para textos relativamente curtos. Esses modelos normalmente limitam a entrada em $n = 512$ tokens/palavras devido ao custo $O(n^2)$ relacionado ao mecanismo de atenção, que dificulta a sua utilização para classificação de textos longos (AINSLIE et al., 2020). Entretanto, existem muitas aplicações de processamento de linguagem natural que utilizam grandes blocos de texto. Um exemplo é a identificação de tópicos de conversas faladas (HAZEN, 2011); (KESIRAJU et al., 2016); (PAPPAGARI et al., 2018) e a predição da satisfação do cliente em call centers (CHOWDHURY et al., 2016); (LUQUE et al., 2017); (PARK; GATES, 2009); (MEINZER et al., 2017).

No caso das conversas de *call center*, a entrada pode variar de bate-papos curtos até conversas longas envolvendo tentativas de resolver problemas complexos da experiência do cliente, resultando em ligações de uma ou mais horas. Um sistema automático de reconhecimento de voz (ASR do inglês *Automatic Speech Recognition*) transcreve essas ligações. Essa

transcrição pode exceder o tamanho de 5000 palavras. Portanto, ETC (AINSLIE et al., 2020) e *Longformer* (BELTAGY et al., 2020) foram propostos e obtiveram resultados no estado da arte através de um balanceamento entre performance e uso de memória.

A abordagem *Longformer* (BELTAGY et al., 2020) usa um padrão de atenção que combina informações locais e globais enquanto também escala linearmente com o comprimento da sequência. Ele pode executar uma ampla gama de tarefas NLP em nível de documento sem segmentar/encurtar a entrada longa e sem arquitetura complexa para combinar informações entre esses fragmentos, alcançando resultados de ponta na tarefa de modelagem de linguagem em nível de caractere . Da mesma forma, o ETC (AINSLIE et al., 2020) também usa um mecanismo de atenção, mas difere do *Longformer* ao combinar a atenção global-local com codificações de posição relativa e máscara flexível, permitindo codificar entradas estruturadas de maneira semelhante às redes neurais de grafo. Embora essas implementações tenham obtido um bom desempenho ao usar *Transformers*, elas fizeram uso de um grande número de dados de treinamento para alcançar seus resultados. (SUN et al., 2019) apresentou uma abordagem que utiliza fragmentação e frações de texto para permitir o uso do BERT em textos longos.

Já (LIN, R. et al., 2022) apresenta um modelo de fusão de recursos multinível baseado no BERT, que é adequado para o BERT através da decomposição hierárquica de textos longos. Em seguida, CNN e BiLSTM empilhado (*Bidirectional Long Short-term Memory*), baseado no mecanismo de atenção, são usados para capturar recursos locais e contextuais do texto, respectivamente. Finalmente, vários recursos são emendados para a tarefa de classificação.

(JING, 2019) desenvolveu um novo tipo de RNN baseado em mecanismo *self-attention* + LSTM (SATT-LSTM), para melhorar o tratamento de sentenças longas e documentos inteiros. Nesse sentido, é utilizada uma LSTM como modelo básico e a sentença não-classificada como entrada. Então, a LSTM codifica a sentença em vários passos de tempo e obtém um vetor de contexto implícito para cada passo, adicionando uma parte de *self-attention* global no modelo LSTM.

### 3.2.3 Aprendendo com Poucos Dados de Treinamento

Em 1998, o mecanismo *"Once learning"* foi proposto para agrupamento de imagens por um exemplo para simular o comportamento humano de aprendizado (WEIGANG, 1998) usando *Self-Organization Map* (SOM). Este paradigma também foi aplicado para identificar as imagens de Radar (WEIGANG; SILVA, N. C. da, 1999). Os pesquisadores (MILLER et al., 2000) definiram um processo para aprender com um exemplo por meio de densidades compartilhadas em transformações. Este método utilizou "conhecimento prévio" para desenvolver um classificador baseado em apenas um único exemplo de treinamento para cada classe. Li FeiFei e outros (FEI-FEI et al., 2003) desenvolveram *"One-shot learning"* para usar conhecimento sobre categorias de objetos para classificar novos objetos como os humanos

fazem. A partir daí, o conceito foi generalizado como *Few-shot Learning* (“Aprendizado em poucos tiros”), aceito pela comunidade e aplicado com sucesso em aplicações de PLN (BROWN et al., 2020).

### 3.2.4 Transferência de Aprendizado

Outra maneira que viabiliza o treinamento de modelos com poucos dados é a Transferência de aprendizado. O uso de aprendizado por transferência em tarefas de PLN não é novo (PAN; TSANG et al., 2011); (DO; NG, 2005) e tem alcançado excelentes resultados ao longo dos anos.

A ideia principal é transferir conhecimento de diferentes domínios de origem para um domínio de destino. Uma abordagem comum para isso é usar representações de vetores de palavras (RUDER et al., 2019; RAINA et al., 2007), como as *Word Embeddings*.

É melhor usada quando dados de treinamento suficientes estão disponíveis apenas em outro domínio de interesse. Nesse caso, a transferência de conhecimento poderia melhorar significativamente o desempenho da aprendizagem, evitando esforços caros de rotulagem de dados (PAN; YANG, Q., 2010).

O BERT (DEVLIN et al., 2018) ainda é apresentado como o estado da arte para a maioria das tarefas de PLN (BULK et al., 2022). Para processar documentos longos, o BERT trunca o texto no tamanho máximo de entrada (1024 para BERT grande). No entanto, essa rede neural geralmente funciona apenas quando informações amplas estão disponíveis no conjunto de dados (KONTONATSIOS et al., 2020; DINTER et al., 2021).

## 3.3 Resumo do Capítulo

A partir da análise da literatura e do trabalho anterior, verificou-se que existem diversos modelos que fazem uso de aprendizagem profunda alcançando ótimos resultados na classificação de texto. Pode-se observar também a presença de técnicas eficientes que contornam os problemas de poucos dados e textos longos.

Destaca-se a eficiência dos modelos DNN, CNN e LSTM no processamento de linguagem natural para a realização da tarefa de classificação. Além disso, a transferência de aprendizado também se mostrou importante para auxiliar na questão de pouca quantidade de dados de treinamento. Os modelos envolvendo *Transformers* também se mostraram as melhores opções para esses tipos de tarefa, porém muitos deles são limitados quanto ao tamanho das sentenças e, principalmente, necessitam de alto poder computacional para serem executados.

Tendo em vista os recursos limitados do projeto, optou-se então por realizar testes utilizando técnicas de representação textual de *Word Embeddings* e os modelos de classifi-

cação DNN, CNN e LSTM. Realizou-se também testes com transferência de aprendizado com *Word2Vec* e *Longformer*. A fundamentação teórica no Capítulo 4 detalha como essas soluções funcionam.

# 4 Fundamentação Teórica

O Aprendizado de Máquina *(Machine Learning)* é o termo usado para definir um sistema que modifica seu comportamento com base em sua própria experiência de aprendizado. Nesse contexto, o processamento de linguagem natural (PLN) relaciona a linguística humana com a linguagem computacional. Dessa forma o PLN possibilita a leitura e interpretação de textos, identificando contextos importantes. Nesta seção, serão abordadas as principais metodologias e técnicas utilizadas em PLN desde a aquisição dos dados, passando pelo pré-processamento e classificação, até avaliação dos resultados, além das ferramentas e estratégias utilizadas para a implementação do projeto.

## 4.1 Mineração de Texto

Os textos não estruturados são as informações mais amplamente disponíveis na internet. Por mais que sejam fáceis de compreendidos pelos humanos, eles são um desafio para as máquinas. Portanto, há a necessidade de desenvolver algoritmos e métodos capazes de processar grandes quantidades de texto para vários tipos de aplicações (ALLAHYARI et al., 2017). Os dados podem ser obtidos através de técnicas de extração em formação e mineração de dados (HAN et al., 2012).

A mineração de texto consiste no processo de extrair informação de qualidade a partir de dados textuais estruturados (através de um banco de dados), semiestruturados (como XML ou JSON), ou não estruturados (como arquivos de texto, vídeos e imagens) (ALLAHYARI et al., 2017).

No contexto do projeto, a informação textual é extraída de plataformas web. Como qualquer página web, essas informações estão organizadas em uma estrutura HTML *(Hypertext Markup Language)* que varia de acordo com a plataforma. Essas informações podem ser extraídas utilizando bibliotecas, como a *Beautiful Soup* (RICHARDSON, 2022), que buscam por estruturas específicas na plataforma e extraem as informações textuais contidas.

As principais dificuldades em lidar com o texto extraído são problemas de *encoding*, já que cada *site* pode utilizar um tipo diferente, e possíveis ruídos representados como markups não tratados inseridos ao longo do texto. Outro problema relevante para o contexto do projeto é de que a extração de dados de diversas plataformas resulta em um conjunto de dados não uniforme com dados não estruturados, onde não está claro onde a informação mais relevante está.

## 4.2 Pré-processamento de Texto

O pré-processamento de texto consiste na aplicação de técnicas a um texto de maneira a construir um vocabulário otimizado que possa ser acoplado a um modelo de maneira mais eficiente. Ele normaliza o texto a partir de algum critério definido e é capaz de reduzir o tamanho do vocabulário ao remover elementos pouco relevantes e simplificando o conteúdo pela redução de variações nas palavras.

Um vocabulário que possui diversas variações da mesma palavra é mais custoso de ser compreendido já que, quando existe necessidade de aprender a contextualização de palavras, cada variação é entendida pelo computador como uma palavra diferente. A redução dessas variações também auxilia na minimização da perda de contexto no texto.

### 4.2.1 Remoção de Caracteres

A remoção de caracteres especiais (&$#), acentuações em caracteres, pontuações (!.?,") e emojis é comum para reduzir o texto sem perder o sentido ou o contexto. Ela também ajuda a remover ruídos que podem estar presentes no texto, como problemas na codificação do texto e markdown desnecessário. A Tabela 4.1 apresenta um exemplo deste processo.

Tabela 4.1 – Exemplo de remoção de caracteres

| | |
|---|---|
| **Original** | Já é de manhã? Não acredito que perdi o nascer do Sol. |
| **Pré-processado** | Ja e de manha Nao acredito que perdi o nascer do Sol |

### 4.2.2 Remoção de Stopwords

Outra técnica comum é a de remover palavras menos significativas chamadas *stopwords*, que ajudam apenas na gramática de algumas palavras, mas não influencia no significado principal desta. Na Tabela 4.2 é possível visualizar um exemplo desta técnica.

Tabela 4.2 – Exemplo de remoção de *stopwords*

| | |
|---|---|
| **Original** | O passado foi o que pensamos quando aconteceu os eventos. Mas o futuro, faria pensar o que seria. |
| **Pré-processado** | passado foi que pensamos quando aconteceu eventos. Mas futuro, faria pensar que seria. |

### 4.2.3 Padronização / Normalização do Texto

Esta técnica tem como objetivo padronizar os caracteres das palavras sob a mesma fonte ou contexto. (ex.; De: Sol, sOl, $ol; Para: sol). O objetivo é não ter a mesma palavra

escrita com caracteres diferentes, como variações maiúsculas ou minúsculas. A Tabela 4.3 apresenta um exemplo da técnica de padronização.

Tabela 4.3 – Exemplo de normalização do texto

| | |
|---|---|
| **Original** | Já é de manhã? Não acredito que perdi o nascer do $ol. |
| **Pré-processado** | já e de manhã? nao acredito que perdi o nascer do sol. |

### 4.2.4 Expansão de Contrações

Em algumas línguas como o inglês, a palavra pode ser abreviada e contraída. É interessante utilizar as palavras na sua forma expandida para a remoção de variações e padronização em que o algoritmo possa identificar o contexto do texto. Na Tabela 4.4 é possível conferir o emprego desta técnica.

Tabela 4.4 – Exemplo de expansão de contrações

| | |
|---|---|
| **Original** | Y'all can't expand contractions I'd think! You wouldn't be able to. How'd you do it? |
| **Pré-processado** | You all cannot expand contractions I would think! You would not be able to. How did you do it? |

### 4.2.5 Stemização

A stemização é o processo de padronizar as palavras reduzindo-as a seu tronco, base ou raiz, removendo as inflexões da palavra. Os textos que recebem esse tratamento podem ser mais difíceis de serem compreendidos por humanos pelo motivo de gerar termos que não são palavras reais. Porém, por conseguir agrupar uma grande quantidade de variações, o computador fica mais eficiente em identificar o significado de cada uma. Um exemplo da aplicação desta técnica é apresentado na Tabela 4.5.

Tabela 4.5 – Exemplo de stemização

| | |
|---|---|
| **Original** | O passado foi o que pensamos quando aconteceu os eventos. Mas o futuro, faria pensar o que seria. |
| **Pré-processado** | O pass foi o que pens qu acontec os event. Mas o futur, far pens o que ser. |

### 4.2.6 Lematização

A lematização é o processo de padronizar as palavras na sua forma base e agrupar no mesmo sinônimo da palavra, alguns exemplos são: contexto gramatical ignorado para sua

forma original (pessoas -> pessoa); verbos normalizados para o tempo presente (aconteceu -> acontece); sinônimos unificados (melhor -> bom). O processo tende a ser mais custoso que a stemização por depender de um dicionário para a realizar a conversão, porém, diferentemente dela, os termos produzidos são palavras reais e podem ser facilmente compreendidas por humanos. É possível visualizar o emprego desta técnica no exemplo exposto na Tabela 4.6.

Tabela 4.6 – Exemplo de lematização

| | |
|---|---|
| **Original** | O passado foi o que pensamos quando aconteceu os eventos. Mas o futuro, faria pensar o que seria. |
| **Pré-processado** | O passado é o que pensa quando acontece o evento. Mas o futuro, faz pensa o que é. |

### 4.2.7 Tokenização

A tokenização é o processo de transformar o texto em uma lista de elementos do vocabulário. Ela transforma caracteres ou palavras, chamados *tokens*, utilizando algum critério de separação como espaços em branco e pontuação. É possível observar o uso desta técnica no exemplo apresentado na Tabela 4.7. A tokenização é importante quando deseja-se inferir informação para cada token de maneira a permitir a compreensão da linguagem.

Tabela 4.7 – Exemplo de tokenização

| | |
|---|---|
| **Original** | Já é de manhã? Não acredito que perdi o nascer do Sol. |
| **Pré-processado** | [“Já”, “é”, “de”, “manhã”, “?”, “Não”, “acredito”, “que”, “perdi”, “o”, “nascer”, “do”, “Sol”, “.”] |

## 4.3 Representação dos Dados

Os dados textuais precisam ser transformados de maneira a permitir seu entendimento pela máquina. Existem várias maneiras de representar esses dados variando em complexidade, desde considerar um número único para cada palavra, até grandes mecanismos de *encoder-decoder* que visam compreender o relacionamento entre as palavras para que possam ser representadas de maneira a reter seu significado e informação contextual.

### 4.3.1 One-hot Encoding

A contextualização dos *tokens* de um vocabulário em formato *one-hot-encode* transforma os tokens em seus índices da ordem do dicionário ou em um vetor do tamanho do vocabulário. Os vetores de índices são representados como valor 1 (um) na respectiva casa do vetor, e o resto igual a zero. Um vocabulário do tipo: { "Quando": 1, "olhei": 2, "você": 3}

através da técnica de *one-hot-encode* representará a palavra "olhei"como o número 2, ou um vetor do tipo [0, 1, 0].

Apesar de ser bem-sucedido na conversão de sentenças para informações numéricas de fácil entendimento pelo computador o *one-hot-encoding* não associa informação às palavras e, portanto, é uma das maneiras mais simples e rápidas de representar os dados.

### 4.3.2 Bag of Words

*Bag of Words* (BOW) é um método de classificação que extrai dados textuais através da criação de dicionários de palavras exclusivas que ocorrem em todos os textos utilizados como dados de entrada do processo. A ideia é quantificar a ocorrência de cada palavra e representá-las em um histograma para realização de análises, como identificar a palavra mais ocorrida e sua relação com outras.

Este método desconsidera o contexto e a ordem em que as palavras ocorrem, resultando na perda do sentido semântico das frases. Outra característica é que a presença de palavras similares em textos diferentes é classificada de maneira isolada, mesmo que estas possuam o mesmo sentido semântico. Ademais, algumas palavras de conexão, como artigos, preposições, conjunções e pontuações, precisam de tratamento dedicado.

O tratamento é realizado por meio da Tokenização (Subseção 4.2.7), processo pela qual ocorre a quebra de uma sequência de caracteres em pedaços menores, podendo ser palavras, números, frases, símbolos ou outros elementos denominados tokens. Durante este processo, os caracteres especiais e de pontuações são removidos, além das palavras de conexão, com o intuito de limpar a base de textos.

Esta abordagem não é a mais adequada para classificação de textos muito grandes, pois pode demandar muito tempo de execução para quantificar cada palavra/vetor. Sua aplicação ocorre principalmente em PLN e classificação de documentos.

### 4.3.3 TF-IDF

O *Term Frequency-Inverse Document Frequency* (TF-IDF) é uma técnica que quantifica a probabilidade de as palavras de um vocabulário estarem presentes entre vários documentos de texto, ponderando se a palavra é frequente ou rara entre estes (SAMMUT; WEBB, 2010).

Este modelo cria um vetor de quantificação para cada palavra do vocabulário, codificando cada documento como um vetor do vocabulário, que representa a estatística dada por TF-IDF. O cálculo deste é feito pela função $TF\_IDF$ dada por:

$$TF\_IDF = TF(w,d) \times IDF(w) \tag{4.1}$$

onde $TF(w,d)$ representa a frequência de uma palavra $w$ entre as que constituem o vocabu-

lário $d$ no documento analisado, ou:

$$TF(w,d) = \frac{\text{Número de vezes que } w \text{ aparece em } d}{\text{Número total de termos em } d} \tag{4.2}$$

e $IDF(w)$ representa em quantos documentos aparece certa palavra, ou:

$$IDF(w) = log\left(\frac{\text{Número total de documentos}}{\text{Número de documentos contendo } w}\right) \tag{4.3}$$

Este método é bastante utilizado para classificação, ou como fator de ponderação para buscas de recuperação de informação, mineração de texto e modelagem de usuários.

### 4.3.4 Word Embeddings

*Word Embeddings* é uma técnica de representação de dados onde cada palavra ou frase é mapeada para um vetor de dimensão $N$. Ela tenta resolver o problema enfrentado pelas técnicas de *weighted-words* (*Bag-of-words*, TF-IDF, etc.) de maneira a tentar capturar o significado semântico das palavras, uma vez que várias palavras são frequentemente utilizadas no mesmo contexto (Exemplo: *"airplane", "aeroplane", "plane"* e *"aircraft"*).

#### 4.3.4.1 Word2Vec

*Word2Vec* (MIKOLOV; SUTSKEVER et al., 2013), utiliza redes neurais rasas para criar um vetor de múltiplas dimensões para cada palavra. Mikolov propôs os modelos CBOW e *Skip-gram*.

O CBOW, do inglês *Continuous Bag-of-Words*, ilustrado na Figura 4.1a, tenta encontrar uma palavra baseado em palavras próximas na frase.

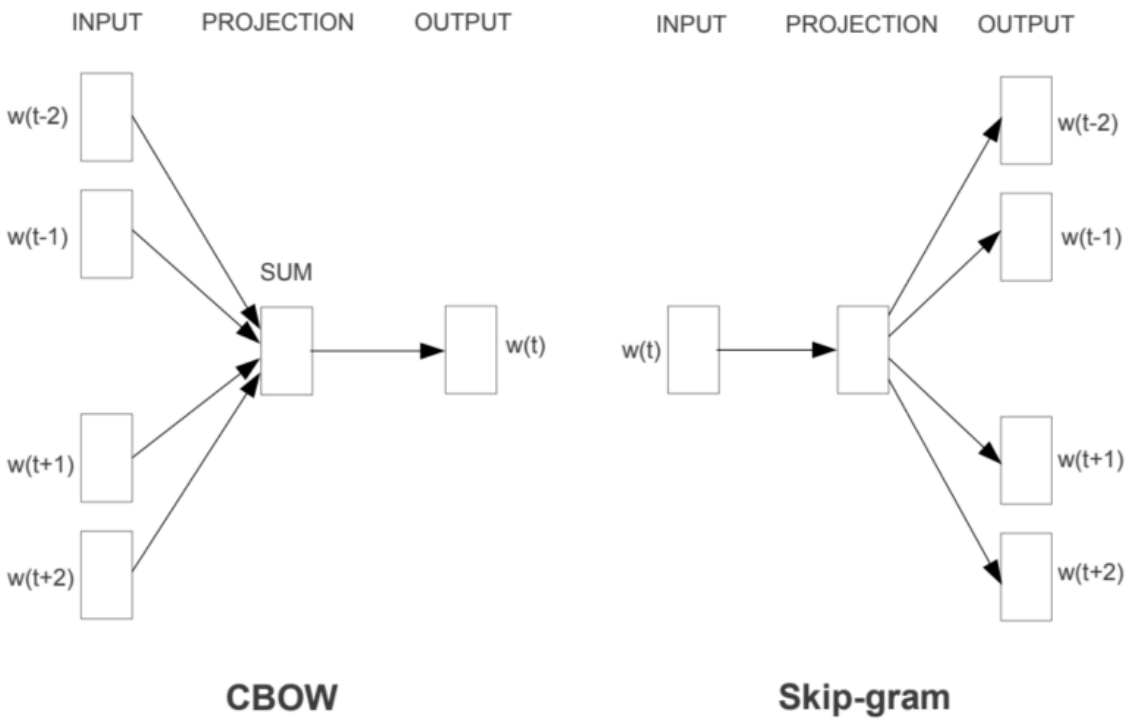
Já o *Skip-gram*, Figura 4.1b, tenta, a partir de uma palavra, prever as palavras da vizinhança, seguindo o modelo de probabilidade descrito na Equação 4.4. (GOLDBERG; LEVY, 2014)

$$arg\,max_\theta \prod_{w \in Text} \left[ \prod_{c \in C(w)} p(c|w;\theta) \right] \tag{4.4}$$

onde $C(w)$ é o conjunto de contextos da palavra $w$ , $Text$ é o corpus, $p(c|w;\theta)$ é a probabilidade condicional do contexto $c$ dada a palavras $w$, e $\theta$ é parâmetro de que maximiza essa probabilidade. Esse método é uma ferramenta muito poderosa para encontrar relacionamentos entre os textos e similaridade entre duas palavras. (GOLDBERG; LEVY, 2014)

#### 4.3.4.2 GloVe

Outra técnica poderosa é a de vetores globais (GloVe - Global Vectors) (PENNINGTON et al., 2014) que se assemelha ao *Word2Vec* onde cada palavra é representada por um




(a) (b)

Figura 4.1 – (a) Arquitetura *Continuous Bag-of-Words*; (b) Arquitetura *Skip-gram*. Fonte: (MIKOLOV; SUTSKEVER et al., 2013)

vetor de múltiplas dimensões treinado baseado nas palavras vizinhas em um *corpus* muito grande. O GloVe constrói uma matriz de contexto de palavras ou de co-ocorrência de palavras explícitas, usando estatísticas de todo o corpus de texto. Essa matriz de contexto é utilizada para treinar *embeddings* de *n* dimensões, permitindo o ajuste para o vetor denso de pesos.

## 4.4 Classificação de Texto

A classificação de texto é um processo de reconhecimento por padrões que se dá através do aprendizado de máquina e Processamento de Linguagem Natural (PLN), sendo estas, vertentes da inteligência artificial que objetivam conferir às máquinas a capacidade de aprender de maneira contínua e realizar previsões baseadas nos dados fornecidos. Neste caso, os dados de entrada são relativos à recursos textuais.

O problema da classificação de texto é centrado em, dado um texto de alguma categoria, a qual conjunto predefinido de classe ele pertence (RUSSELL; NORVIG, 2010).

Para alcance de tal objetivo, existem diferentes técnicas conceituadas e empregadas na academia. Esta seção apresenta as mais populares.

### 4.4.1 Naive Bayes

*Naive bayes* é um modelo de classificador probabilístico com ótimo desempenho para uma gama de aplicações, sendo um dos algoritmos de aprendizado de uso geral mais eficazes. Este método é baseado no teorema de Thomas Bayes, e toma como premissa a suposição de independência condicional mútua entre as características sendo analisadas no modelo. Desta forma, para classificação textual, as possíveis dependências estatísticas entre tokens são desconsideradas.

O teorema de Bayes estabelece que, para cada vetor $x_1, ..., x_n$ de características, a probabilidade de uma classe $C$ é dada por:

$$P(C|x_1, ..., x_n) = \frac{P(C)P(x_1, ..., x_n|C)}{P(x_1, ..., x_n)} \tag{4.5}$$

Agora considerando a suposição de independência condicional mútua, o modelo pode ser expresso pela Equação 4.6:

$$P(C|x_1, ..., x_n) \propto P(C) \prod_{i=1}^{n} P(x_i|C) \tag{4.6}$$

onde $\propto$ denota proporcionalidade, $P(C)$ é a probabilidade da classe $C$ e $P(x_i|C)$ é a probabilidade condicional da característica $x_i$ ocorrer dada a ocorrência de $C$, com $i$ variando de 1 à $n$. Assim, uma previsão determinística pode ser obtida escolhendo a classe mais provável (RUSSELL; NORVIG, 2010).

O *Naive bayes* é um modelo adequado para problemas grandes, pois, para $n$ atributos booleanos, existem apenas $2n + 1$ parâmetros e nenhuma pesquisa se faz necessária para encontrar a hipótese de máxima probabilidade de *bayes*. O modelo também não tem dificuldade com a ausência de dados, ou mesmo com dados ruidosos (ZHANG, H., 2004).

### 4.4.2 Support Vector Machines

O *Support Vector Machines* (SVM) é uma técnica de classificação linear binária, utilizada em casos onde é possível plotar e dividir um conjunto de dados por uma linha reta. Tais conjuntos são designados linearmente separáveis. Para esse propósito, o algoritmo do SVM busca por fronteiras de decisão, ou hiperplanos, que melhor dividem os dados baseado nas classes impostas. A escolha do melhor hiperplano é feita analisando a assertividade e a distância entre dados próximos.

Quando o conjunto de dados não é linearmente separável, o SVM utiliza-se da função algébrica *Kernel* para aumentar a dimensão do problema e aplicar transformações no conjunto de dados, possibilitando desta maneira o traçado de um novo hiperplano.

Esta abordagem é adequada para espaços multidimensionais, apresentando boa assertividade, e ainda apresenta o conveniente de poder aplicar função *Kernel* quando os

dados não se enquadram no critério. Além disso, ainda se mostra eficaz quando o número de características das classes são maiores que o número de amostras.

### 4.4.3 Random Forest

O *Random Forest* é um modelo classificador supervisionado composto por um número massivo de árvores de decisão que objetivam diminuir a possibilidade da ocorrência de *overfitting*, ou sobreajuste. O *overfitting* é um problema que ocorre quando, ao se utilizar uma única árvore de decisão, seu modelo se ajusta adequadamente a uma base de dados, porém, apresenta dificuldades para se ajustar às demais amostras não previstas, impossibilitando sua generalização.

Neste modelo, as árvores são criadas a partir de subamostras, permitindo que as árvores se divirjam quanto à relevância e quantidade de características consideradas, resultando assim em maior generalização do modelo. A decisão final do modelo é tomada pela soma das decisões individuais de cada árvore, diminuindo a influência de árvores enviesadas.

### 4.4.4 Aprendizado Profundo

Aprendizado profundo é um ramo do aprendizado de máquina (*machine learning*) que busca simular computacionalmente as interconexões dos neurônios do cérebro, para conferir às máquinas a capacidade de aprender.

Estes sistemas são constituídos com recursos hardware e software que simulam os padrões de pensamento do cérebro humano, “aprendendo” padrões a partir de uma quantidade massiva de dados, aplicando filtros, estabelecendo relações, construindo modelos e corrigindo os próprios erros.

Uma rede neural artificial apresenta uma grande quantidade de nós que interagem continuamente uns com os outros e são divididos em camadas. Quanto mais camadas uma rede possuir, mais complexa ela se torna.

#### 4.4.4.1 Perceptron Multicamadas

O *perceptron* é um modelo de classificação linear proposto matematicamente por (MCCULLOCH; PITTS, 1943) em 1943, e implementado computacionalmente por Frank Rosenblatt, em 1958 (BENTO, 2021). Também conhecido como neurônio digital ou neurônio de McCulloch-Pitts, o *perceptron* é uma rede neural de camada única, capaz de receber vários dados de entrada e produzir uma única saída binária.

A saída do *perceptron* é determinada pela soma ponderada da entrada $X_j$ associada a um peso $W_j$, com $j$ variando de 1,..., $n$, sendo $n$ a quantidade total de entradas. A classificação é determinada com base em algum limiar *(threshold)* (BOOK, 2022b), como se segue na Equação 4.7.

$$output = \begin{cases} 0, & \text{se } \sum_{j=1}^{n} W_j X_j \leq threshold \\ 1, & \text{se } \sum_{j=1\ n} W_j X_j > threshold \end{cases} \tag{4.7}$$

O *perceptron* é capaz de lidar apenas com dados linearmente separáveis, portanto, é um modelo mais simples de classificação. Para lidar com classificações mais complexas, considerando mais parâmetros na tomada de decisão, o conceito de multi-camada foi introduzido.

O *perceptron* multicamadas (do inglês *multilayer perceptron* - MLP) é um modelo de classificação formado pela generalização de *perceptrons* dispostos em mais de uma camada. A partir deste momento, o modelo já é considerado por alguns autores, como (KOWSARI et al., 2019; GU et al., 2014), uma rede neural artificial profunda. No MLP existem três tipos de camadas: a camada de entrada, responsável por receber o sinal de entrada a ser processado, a camada oculta (podendo ser mais de uma) que realiza a maior parte do processamento, propagando as combinações lineares das camadas anteriores para as seguintes, e por fim, a camada de saída que realiza a tarefa de classificação final (BENTO, 2021).

Para possibilitar a capacidade de aprendizagem à rede neural MLP, ela conta com o recurso de retropropagação (do inglês *backpropagation*), que realiza o ajuste dos pesos ao longo rede, de modo a reduzir o erro entre a saída obtida e a saída desejada.

#### 4.4.4.2 SNN

As SNNs (do inglês *Shallow Neural Networks*, ou Redes Neurais Rasas) são as redes neurais mais simples. Elas são chamadas de “rasas” porque possuem apenas uma ou duas camadas ocultas. Em função disso, ela normalmente possui limitações relativas à complexidade dos problemas que é capaz de ser treinada para resolver. Por apresentar uma estrutura simplificada, a SNN apresenta maior velocidade de processamento, sendo possível validar alterações no modelo treinado de maneira mais rápida.

#### 4.4.4.3 DNN

A DNN, do inglês *Deep Neural Networks*, é uma rede neural profunda com mais camadas ocultas. As várias camadas de abstração proporcionam às redes profundas uma vantagem convincente em aprender a resolver problemas complexos de reconhecimento de padrões (NIELSEN, 2015). DDNs são compostos de camadas conectadas onde cada camada recebe conexões da camada anterior e fornece conexões para o seguinte.

Sua implementação usa um algoritmo padrão de retropropagação. A camada de saída possui o número de classes a serem classificadas, apenas uma para classificação binária, e é uma função *softmax* (KOWSARI et al., 2019). O objetivo é aprender o relacionamento entre entrada e saída utilizando as camadas ocultas conforme mostrado na Figura 4.2.

De modo geral, o DNN é uma classe genérica de rede neural profunda que abrange todos os tipos de redes que apresentam em seu modelo várias camadas ocultas de processamento. Desta forma, o MLP, o CNN (apresentado na Subseção 4.4.4.4 seguinte) e o LSTM (apresentado na Subseção 4.4.4.6) são exemplos de DNN.

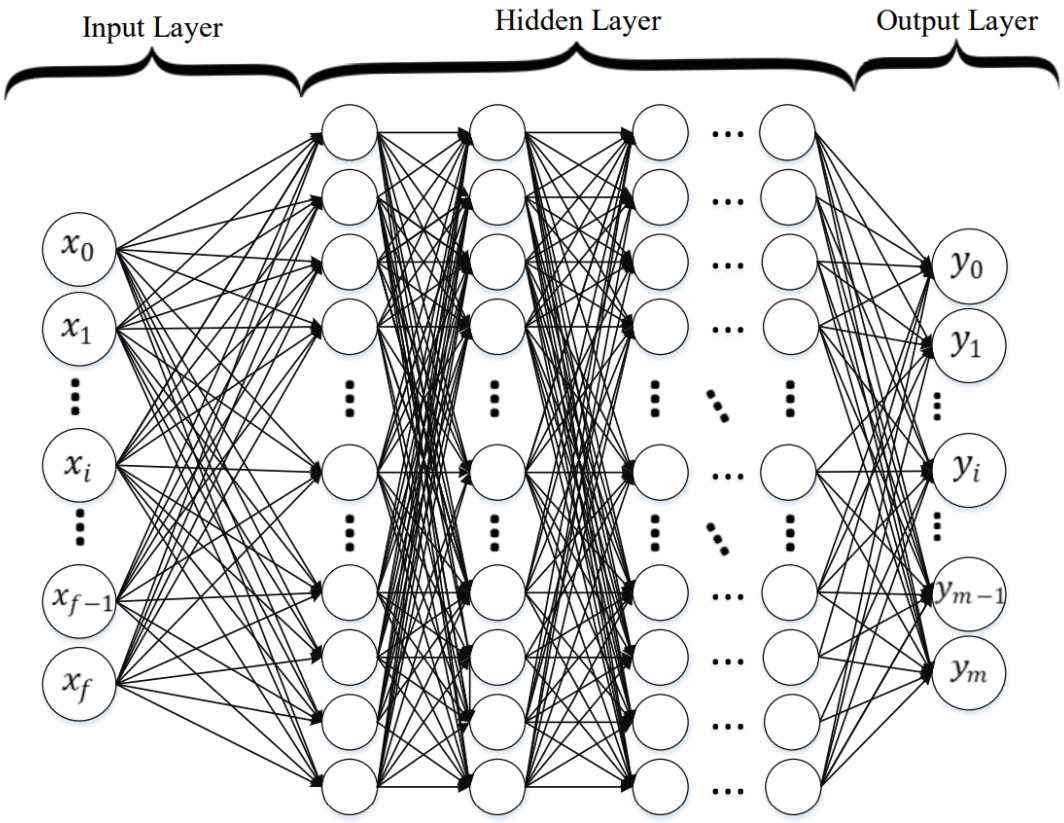



Figura 4.2 – Exemplo de rede neural profunda (DNN, do tipo MLP) totalmente conectada. Fonte: (KOWSARI et al., 2019).

#### 4.4.4.4 CNN

A Rede Neural Convolucional (CNN) é uma rede de aprendizado profundo, inicialmente desenvolvida para processamento de dados em formato de grade, como imagens. A CNN é composta por três tipos de camadas: a camada convolucional, responsável pela processamento massivo dos dados com auxílio de filtros e mapas de recursos; a camada *pooling*, responsável por realizar agrupamento de dados com o objetivo de reduzir a complexidade computacional, diminuindo a quantidade de saídas da rede de maneira que seja preservado características importantes; e camada totalmente conectada, que realiza a tarefa de classificação baseada nas características extraídas de camadas anteriores.

Um problema associado ao uso de CNN para classificação de textos é o elevado número de canais (espaço de recurso). Se comparado com a sua aplicação para classificação de imagens, que geralmente apresentam 3 canais de RGB, a CNN aplicada a classificação de textos pode apresentar um número muito elevado de canais, na ordem de dezenas de milhar.

Embora seja mais comumente conhecido por aplicações de processamento de imagem (2D) e visão computacional (2D a 3D), a convolução também é aplicada em 1D e tem alcançado ótimos resultados em classificação de texto (KIM, 2014).

#### 4.4.4.5 RNN

Rede neural recorrente (RNN) é um tipo de rede neural artificial que possui um comportamento dinâmico temporal. Ela possui um estado interno (memória) que pode ser utilizado para processar sequências de comprimento variável de entradas. Uma grande vantagem do modelo é a de compartilhar recursos aprendidos em diferentes posições de texto que não podem ser obtidos em uma rede neural padrão (MIKOLOV; KARAFIÁT et al., 2010).

#### 4.4.4.6 LSTM

LSTM, do inglês *Long Short-Term Memory*, é um tipo de rede neural recorrente (RNN) que tem como principais características a capacidade de aprender dependências de longo prazo (JUNIOR, 2019) e preservar memórias de curto prazo. Com isso, as redes LSTM contornam de maneira eficaz o problema do gradiente de fuga (PASCANU et al., 2012) e se tornam boas candidatas para aplicações em que o contexto prévio da informação de entrada é importante para determinar a resposta futura, a exemplo de tradutores automáticos e assistentes de voz.

Sua arquitetura é composta de células de memória (Figura 4.3) conectadas entre si de maneira recorrente. Dentro de cada célula encontram-se elementos responsáveis por regular o fluxo da informação de acordo com o peso e a função de ativação que cada um possui. Estes elementos são coletivamente chamados de portas (em inglês, *gates*) e desempenham funções distintas no tratamento das informações, sendo estas:

- **Porta de Entrada:** é o elemento que adiciona informações diretamente à célula de memória. O tratamento da informação é feito em duas etapas, sendo a primeira uma função sigmoide que processa a concatenação da entrada $X_t$ com a saída anterior $h_{t-1}$, onde $t$ representa instante de tempo. A segunda, obtida paralelamente, cria com a mesma entrada um vetor usando a função tangente hiperbólica. Ambos os resultados parciais são então multiplicados para, enfim, gerar o resultado útil (BOOK, 2022a).
- **Porta de Esquecimento:** é o elemento responsável por decidir pela persistência de um valor anterior da saída que a própria célula gerou. Processa a mesma concatenação da entrada ($X_t$) com a saída anterior ($h_{t-1}$), aplicando matrizes de peso e *bias* para gerar um resultado intermediário. Esse resultado então passa por uma função de ativação que fornece uma saída binária, sendo 0 para indicar o esquecimento da informação e 1 para mantê-la para um processamento futuro (BOOK, 2022a).

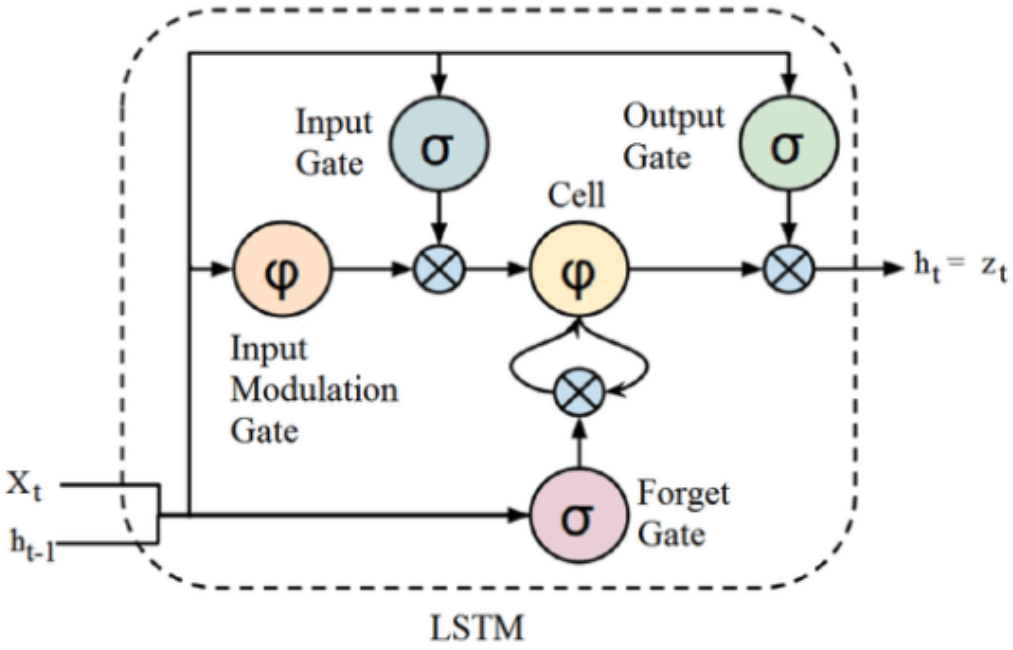



Figura 4.3 – Bloco de memória de LSTM. Fonte: (BOOK, 2022a)

- **Porta de Saída:** Aplica uma regulação por função sigmoide na entrada original ($X_t$) concatenada com a saída anterior ($h_{t-1}$) da célula, e multiplica o resultado com o vetor gerado na saída da Porta de Entrada. Essa multiplicação vira, por fim, o resultado do processamento da própria célula que será passado para a seguinte (BOOK, 2022a).

Assim como o RNN, o LSTM contém uma arquitetura em cadeia que considera informações de nós anteriores, com a diferença de que também contém várias portas que regulam a quantidade de informações permitidas em cada estado do nó. Esse tipo de arquitetura associada a *Word Embeddings* tem obtido excelentes resultados na classificação de textos (WANG, G. et al., 2018; XIAO et al., 2018).

## 4.5 Transformers

O *Transformer* (VASWANI et al., 2017) é um modelo de *deep learning* que objetiva realizar tarefas de NLP através de um mecanismo de *self-attention* (auto-atenção) que computa representações das entradas e saídas sem a utilização de operações de RNN ou Convolução. Ele utiliza a arquitetura codificador-decodificador (*encoder-decoder*) como a mostrada na Figura 4.4.

O codificador tem como responsabilidade gerar codificações que contenham informações sobre quais partes das entradas são relevantes entre si. Ele consiste em uma pilha de

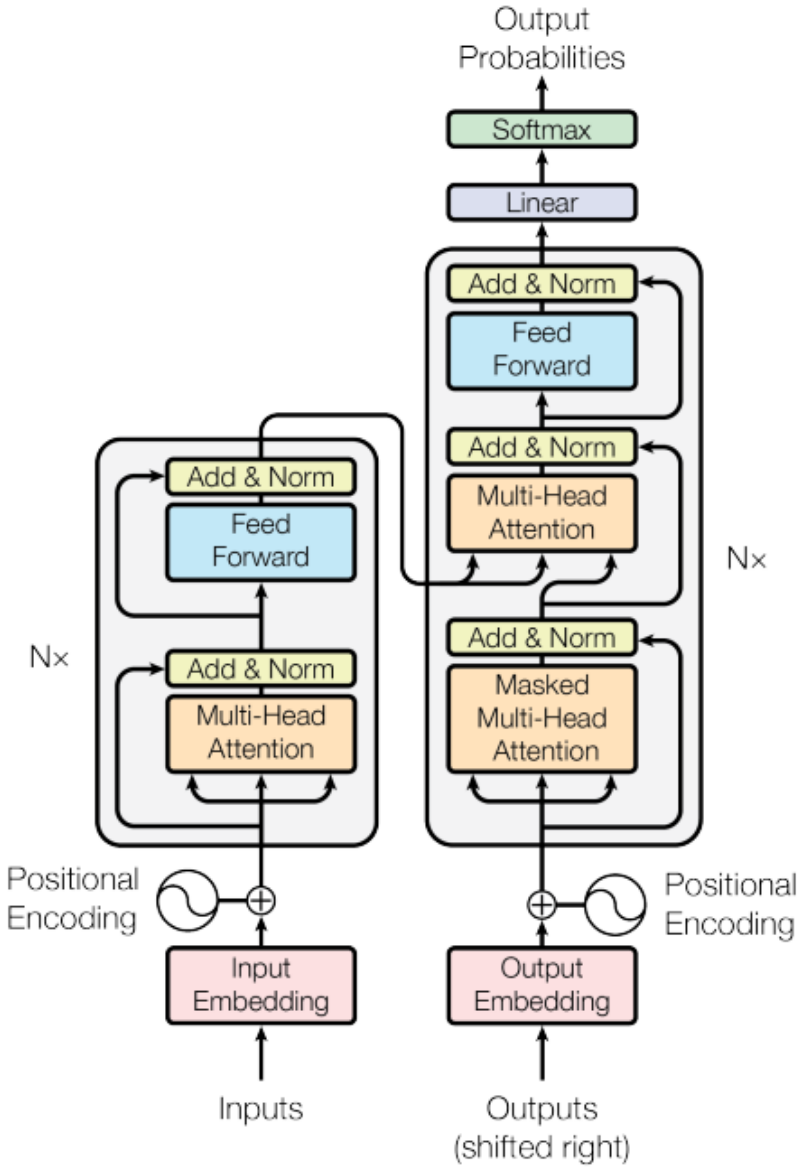



Figura 4.4 – Arquitetura do modelo *Transformer*. Fonte: (VASWANI et al., 2017)

$N = 6$ camadas idênticas, em que cada camada é composta de duas subcamadas.

A primeira delas implementa o mecanismo de *Multi-head self-attention*. Esse mecanismo cria $h$ heads que recebem versões diferentes de consultas *(queries)*, chaves e valores, produzindo $h$ saídas em paralelo que são utilizadas para gerar o resultado final.

A segunda subcamada é uma rede neural totalmente conectada (FFN, do inglês *Feed-forward Network*) que consiste em duas transformações lineares com funções de ativação do tipo *ReLU* (função de ativação linear retificada) entre elas.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{4.8}$$

Cada uma das seis camadas do encoder aplica as mesmas transformações lineares

para todas as palavras de uma frase, mas emprega pesos e *bias* diferentes ($W_1, W_2, b_1, b_2$). Além disso, entre as subcamadas existe uma conexão residual ao redor. Uma camada de normalização sucede a cada subcamada e realiza o papel de normalizar a soma computada entre a entrada $x$ e a saída gerada pela subcamada.

A arquitetura do *Transformer* não captura por padrão informações relacionadas à posições relativas das palavras nas frases e essa informação precisa ser injetada com a introdução de *positional embeddings* aos *embeddings* de entrada. Esses *positional embeddings* possuem as mesmas dimensões da entrada e são gerados com a utilização de funções seno e cosseno de frequências diferentes, e então são somados à entrada para injetar a informação de posição.

Já o decodificador faz o oposto, pegando as codificações e utilizando as suas informações contextuais para gerar uma sequência de saída. Ele também consiste em uma pilha de $N = 6$ camadas idênticas, porém cada uma delas é formada por uma terceira subcamada que realiza atenção *multi-head* em cima da saída da pilha do *encoder*, em adição às duas subcamadas de cada *encoder*. Da mesma maneira do *encoder*, existem conexões residuais ao redor de cada subcamada seguidas por uma camada de normalização. Também existe uma modificação na subcamada de *self-attention* para evitar que posições se atentem a posições subsequentes. Isso, em conjunto com o *offset* de uma posição dos *embeddings* de saída, garante que as predições para uma posição $i$ dependam apenas de saídas conhecidas de posições menores que $i$ (VASWANI et al., 2017).

O mecanismo de *self-attention* relaciona posições diferentes de uma sentença para computar a sua representação. Atualmente sua utilização está atrelada a maior parte dos resultados estado-da-arte em processamento de linguagem natural (WOLF et al., 2020).

### 4.5.1 Longformer

O *Longformer* é um modelo baseado em *Transformers* que foi apresentado por (BELTAGY et al., 2020) como alternativa para lidar com textos longos. Ele resolve uma limitação apresentada pela primeira geração de *Transformers* e arquiteturas baseadas em BERT (citar BERT) que é o tamanho máximo de entrada de 512 *tokens*. A razão por trás dessa limitação é que o mecanismo de *self-attention* escala quadraticamente com o comprimento da sequência de entrada - $O(n^2)$. Devido a essa limitação esses outros modelos necessitam truncar a entrada e isso pode potencialmente resultar em perda de informação relevante.

O *Longformer* soluciona esse problema apresentando uma arquitetura modificada onde a operação de *self-attention* escala linearmente com o comprimento da sentença de entrada. Ele combina uma janela de contexto local de *self-attention* e uma *global-attention* (atenção global) motivada pela tarefa final que codifica um bias indutivo sobre a tarefa.

A Figura 4.5.b mostra o padrão de atenção *"Sliding Window"* que emprega uma janela

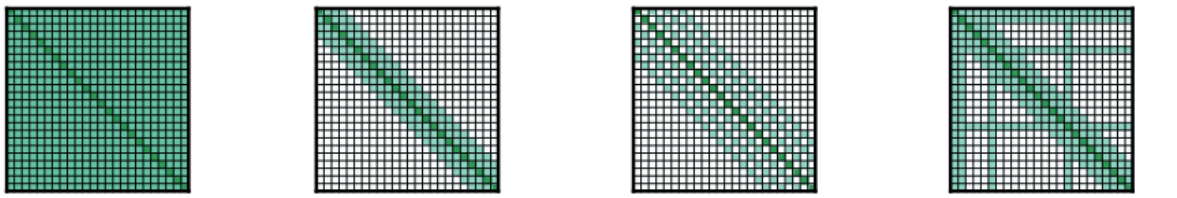



Figura 4.5 – Comparando o padrão de *self-attention* completo e os padrões de atenção do *Longformer* – Fonte: (BELTAGY et al., 2020)

de atenção de tamanho fixo ao redor de cada *token*. O uso desse tipo de atenção permite construir representações que incorporam informações através de toda a entrada, de maneira similar à CNNs (WU et al., 2019). Para aumentar o campo receptivo sem aumentar o custo computacional a janela pode ser dilatada (Figura 4.5.c). (BELTAGY et al., 2020) descobriu que definir configurações diferentes de dilatação por cabeça, na atenção multi-head, melhora a performance ao permitir algumas cabeças focarem no contexto local enquanto outras focam no contexto maior.

A adição da *global-attention* (Figura 4.5.d) em algumas poucas localizações da entrada de maneira simétrica mantém a complexidade em $O(n)$ já que a quantidade de locais é pequena em relação a $n$. Essa *global-attention* permite a adição do *bias* indutivo na atenção do modelo.

O *Longformer* permitiu o processamento de sequências de mil caracteres sem enfrentar o gargalo de memória de arquiteturas tipo BERT e alcançou estado-da-arte em diversos *benchmarks*, com destaque para os 95.7% de acurácia na classificação do *dataset* IMDB (MAAS et al., 2011) e 94.8% de f1-score na classificação do *dataset Hyperpartisan* (KIESEL et al., 2019).

## 4.6 Transferência de Aprendizado

Outra tendência empregada nas tarefas de PLN é o uso de técnicas de *Transfer Learning* que permite usar o conhecimento de um domínio original e fornece um meio de transferir o referido conhecimento para o domínio de destino para aumentar a cobertura de dados. Enquanto mais comumente encontrado em aplicações de processamento de imagens, mostrou-se eficiente em aplicações de PLN ao permitir o compartilhamento de conhecimento como similaridade na representação linguística (RUDER et al., 2019). A transferência de aprendizado pega o conhecimento previamente adquirido no domínio original e continua o treinamento com novos dados.

Dentro do domínio da PNL, tornou-se comum abordar problemas por meio de aprendizado de transferência, em vez de construir um modelo do zero, usando um modelo pré-

treinado em outra tarefa e ajustando-o por meio de treinamento adicional no conjunto de dados de interesse (MOHAMMED; ALI, 2021; CHURCH et al., 2021).

Para classificar os textos disponíveis em uma sobrecarga de conjuntos de dados, foi proposto o uso de modelos de linguagem pré-treinados (PLM). Os PLMs são previamente treinados em um grande *corpus* de texto e têm alguma capacidade de entender o contexto do texto, como modelos baseados em *Transformer* (VASWANI et al., 2017).

O *fine-tuning* ou ajuste fino consiste em um novo treinamento dos pesos, que já haviam sido treinados anteriormente em outro domínio, agora aplicado ao conjunto de dados de interesse. Nesse caso a camada de saída será treinada do zero, enquanto parâmetros de outras camadas herdadas do domínio anterior serão apenas ajustadas com base nos novos dados.

## 4.7 TensorFlow e Keras

O *TensorFlow* é uma plataforma de aprendizado de máquina de código aberto de ponta a ponta. Ela consiste em uma infraestrutura para programação diferenciável capaz de manipular matrizes $N$-dimensionais (tensores) (KERAS, 2022). De acordo com (FCHOLLET, 2020), suas características principais são:

- O *TensorFlow* pode aproveitar aceleradores de *hardware*, como GPUs e TPUs.
- O *TensorFlow* pode calcular automaticamente o gradiente de expressões de tensor diferenciáveis arbitrárias.
- A computação do *TensorFlow* pode ser distribuída para um grande número de dispositivos em uma única máquina e um grande número de máquinas (potencialmente com vários dispositivos cada).

O *Keras* é a API de alto nível do *TensorFlow* e possui uma interface acessível e altamente produtiva para resolver problemas de aprendizado de máquina, com foco em aprendizado profundo (KERAS, 2022). Ela fornece abstrações e blocos de construção que facilitam o desenvolvimento de soluções de aprendizagem de máquina como camadas, modelos, otimizadores, funções de perda, métricas, entre outros.

A camada (classe *Layer*) é a principal abstração do *Keras*. Uma camada encapsula um estado (pesos dos nós) e alguma computação específica para cada tipo de camada. Na Subseção 4.7.1 serão apresentadas algumas das principais camadas utilizadas no projeto.

### 4.7.1 Camadas Keras

- ***Embedding:*** Transforma índices ou inteiros positivos em vetores densos de tamanho fixo definido como parâmetro da camada, assim como o tamanho do vocabulário. Exemplo: [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]].
- ***Dense:*** Consiste na camada de rede neural densamente conectada. Ela possui um tamanho fixo definido como parâmetro. Sua função de ativação pode ser passada através de um parâmetro ou por meio do uso da camada Activation.
- ***Activation:*** Aplica uma função de ativação a uma saída como ReLU, Softmax, step (degrau), etc.
- ***Flatten:*** Achata a entrada, garantindo que o formato de saída seja um vetor unidimensional.
- ***Dropout:*** Esta camada define aleatoriamente algumas unidades de entrada como 0. A taxa de quantas unidades de entrada terão seus valores zerados é definida através de um parâmetro que varia de 0 (nenhum) a 1 (todos). As entradas não definidas como 0 são escaladas em 1/(1 - taxa) de modo que a soma das entradas permaneça inalterada. Esse processo auxilia na prevenção de overfitting e só se aplica durante a etapa de treinamento.
- ***Conv1D:*** Essa camada cria um núcleo que convoluciona com a entrada da camada em uma única dimensão para produzir um tensor de saídas. O tamanho do núcleo e a distância entre saltos de convolução são definidos como parâmetros da camada.
- ***GlobalMaxPooling1D:*** Essa camada realiza um agrupamento máximo global para dados de apenas uma dimensão. Ela reduz a amostra da representação da entrada, obtendo o valor máximo para aquela dimensão.
- **LSTM**: Implementa o mecanismo de LSTM e recebe como parâmetro um inteiro positivo referente à dimensionalidade do espaço de saída.

## 4.8 Métricas para a Avaliação dos Resultados

Para avaliar a performance de um modelo de classificação em aprendizagem de máquina, dentro de um cenário de aprendizagem supervisionada, pode-se fazer uso da matriz de confusão. Uma classificação binária pode ser vista como instâncias de classificação positivas ou negativas. Quando se compara esses valores com a expectativa tem-se os conceitos de Verdadeiro Positivo, Falso Positivo, Verdadeiro Negativo e Falso Negativo. Esses conceitos montam a matriz de confusão apresentada na Figura 4.6.

| | | Valor Predito | |
|---|---|---|---|
| | | **Sim** | **Não** |
| **Real** | **Sim** | Verdadeiro Positivo (TP) | Falso Negativo (FN) |
| | **Não** | Falso Positivo (FP) | Verdadeiro Negativo (TN) |

Figura 4.6 – Matriz de confusão. Fonte: (DIOGO NOGARE, 2020)

Verdadeiros positivos representam as instâncias onde o valor foi predito como positivo e o valor real também. Falsos positivos são aqueles onde o valor predito foi positivo porém o valor real não é. Verdadeiros negativos são os casos onde o valor foi predito como negativo e o valor real é negativo. Já os Falsos negativos representam o cenário onde o classificador previu negativo quando na realidade era positivo. Acima da matriz de confusão é possível é possível realizar o cálculos de das métricas de Acurácia, Precisão, *Recall*, e *F1-score*.

**Acurácia:**
A acurácia $Ac$ representa a quantidade de dados classificados como positivos e negativos corretamente. Ela é calculada com a equação 4.9.

$$Ac = \frac{\text{Verdadeiros Positivos} + \text{Verdadeiros Negativos}}{\text{Verdadeiros Positivos} + \text{Verdadeiros Negativos} + \text{Falsos Positivos} + \text{Falsos Negativos}} \tag{4.9}$$

**Precisão:**
A precisão representa a quantidade positiva classificada corretamente. Ela é calculada com a equação 4.10.

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falso Positivos}} \tag{4.10}$$

**Recall:**
O Recall é a taxa dos valores classificados como positivos comparado com quantos estão corretas. Ele é calculado com a equação 4.11.

$$Recall = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falso Negativos}} \tag{4.11}$$

**F1-score:**
O F1-score é a média harmônica entre Precisão e Recall. Ou seja:

$$\text{F1-score} = \frac{2}{\frac{1}{\text{Precisão}} + \frac{1}{\textit{Recall}}} \tag{4.12}$$

Que se traduz na equação 4.13.

$$\text{F1-score} = \frac{2 \times \text{Verdadeiros Positivos}}{2 \times \text{Verdadeiros Positivos} + \text{Falsos Positivos} + \text{Falsos Negativos}} \tag{4.13}$$

# 5 Descrição dos Datasets

Durante o desenvolvimento deste trabalho foram utilizados três *datasets* para o treinamento dos modelos de aprendizado profundo: o *dataset* Preliminar, o *dataset* PPF e o *dataset* final. Eles possuem diversas colunas as quais a equipe de raspagem acreditou interessantes para uso como *opo_titulo* que contém o título da oportunidade, *opo_tipo* representando o tipo de oportunidade (*grand*, *fellowship*, *scholarship*, etc.), *codigo* sendo um identificador único para a oportunidade, *deadline* que apresenta o prazo de aplicação para a oportunidade, entre outros.

Para a classificação as colunas importantes são *opo_texto* e *opo_texto_ele*, e correspondem ao texto completo da oportunidade e o pedaço do texto com informações de elegibilidade respectivamente. A coluna *opo_texto_ele* foi criada a partir da busca simples por palavras chave e na maior parte das vezes não possui informações que agregam ao entendimento da oportunidade. Alguns exemplos de oportunidades raspadas podem ser observados na Tabela 5.1.

Tabela 5.1 – Exemplos de oportunidades raspadas

| **Elegibilidade** | **opo_texto** | **opo_texto_ele** |
|---|---|---|
| Elegível | Science and technology are at the heart of everything we do, driving innovations that enable us to contribute to a sustainable future. In this endeavor, we are seeking research proposals in the field of Green Chemistry. | Science and technology are at the heart of everything we do, driving innovations that enable us to contribute to a sustainable future. In this endeavor, we are seeking research proposals in the field of Green Chemistry. |
| | | Continua... |

**Tabela 5.1**

| Elegibilidade | opo_texto | opo_texto_ele |
|---|---|---|
| Elegível | The Tinker FoundationÃ¢â‚¬â„¢s Institutional Grants program provides project funding to organizations working to improve the lives of Latin Americans, with an emphasis on support for organizations in the region. | Organization Status The Tinker Foundation provides grants only to organizations that are charitable in nature, i.e., with a United States 501(c)(3) tax status or its equivalent if the organization is located outside the U.S. Organizations from Latin America do not need to have United States 501(c)(3) status. Geographic Focus The project must be focused on the Spanish- and Portuguese-speaking countries of Latin America, including: Argentina Bolivia Brazil Chile Colombia Costa Rica Cuba Dominican Republic Ecuador El Salvador Guatemala Honduras Mexico Nicaragua Panama Paraguay Peru Uruguay Venezuela |

Continua...

**Tabela 5.1**

| Elegibilidade | opo_texto | opo_texto_ele |
|---|---|---|
| Não Elegível | The EFIS and Immunology Letters Short-term Fellowship offered by the European Federation of Immunological Societies (EFIS) and Immunology Letters (IL) provides a funded three month fellowship for early career immunology researchers who are looking for an opportunity to collaborate or train in another European country. What is funded This fellowship provides 1750 EUR per month for three months and up to 500 EUR for travel expenses. Duration Three months of funding for this short-term fellowship. Eligibility Applicants should be a member of an EFIS-affiliated society or working in a European country. The proposed fellowship must be conducted at a European institution in a country other than that of the applicantÃ¢â‚¬â„¢s current residence. Applicants must be no more than 35 years old and at minimum should hold a MasterÃ¢â‚¬â„¢s degree (an extension of this age limit of up to 18 months may be granted for parental leave, civil or military service, etc.). | The EFIS and Immunology Letters Short-term Fellowship offered by the European Federation of Immunological Societies (EFIS) and Immunology Letters (IL) provides a funded three month fellowship for early career immunology researchers who are looking for an opportunity to collaborate or train in another European country. What is funded This fellowship provides 1750 EUR per month for three months and up to 500 EUR for travel expenses. Duration Three months of funding for this short-term fellowship. Eligibility Applicants should be a member of an EFIS-affiliated society or working in a European country. The proposed fellowship must be conducted at a European institution in a country other than that of the applicantÃ¢â‚¬â„¢s current residence. Applicants must be no more than 35 years old and at minimum should hold a MasterÃ¢â‚¬â„¢s degree (an extension of this age limit of up to 18 months may be granted for parental leave, civil or military service, etc.). |

Continua...

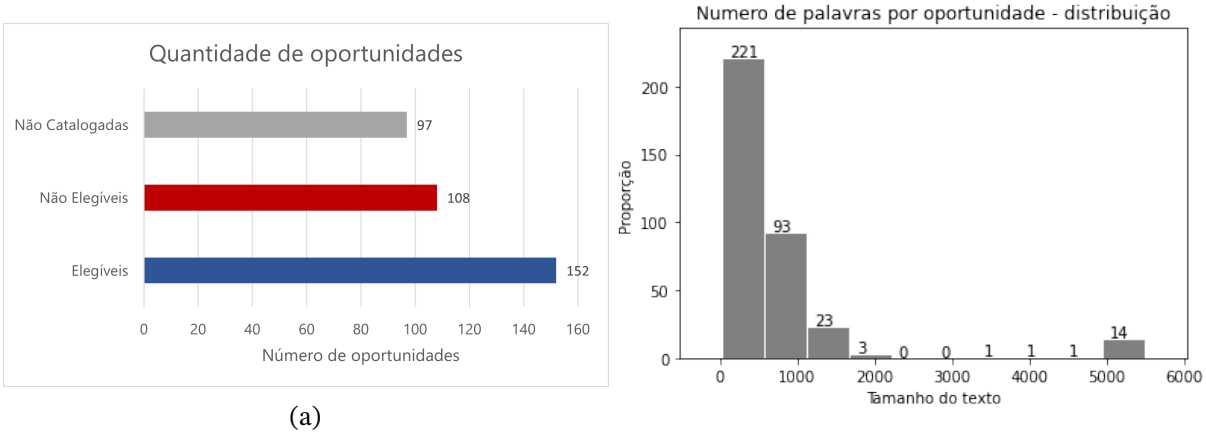
**Tabela 5.1**

| Elegibilidade | opo_texto | opo_texto_ele |
|---|---|---|
| Não Elegível | The Call offers two 12-month grants for Visiting Professors/Researchers who fleed Ukraine. Grants will be allocated until funds are available (rolling application). Total budget: Ã¢â€šÂ¬40,000 Contacts: international.cooperation @ateneo.univr.it | The Call offers two 12-month grants for Visiting Professors/Researchers who fleed Ukraine. Grants will be allocated until funds are available (rolling application). Total budget: Ã¢â€šÂ¬40,000 Contacts: international.cooperation @ateneo.univr.it |

Ao observar a Tabela 5.1 é possível identificar que algumas oportunidades não apresentam explicitamente o critério de elegibilidade. Outro detalhe importante a ser notado é a existência de alguns problemas de codificação que acontecem na hora de raspar o HTML, o que acaba resultando em caracteres estranhos presentes ao decorrer de alguns textos. Também é possível notar que em algumas oportunidades não é possível diferenciar as colunas *opo_texto* e *opo_texto_ele*.

## 5.1 Dataset Preliminar

O *dataset* Preliminar foi disponibilizado pelo grupo de pesquisa responsável por (SILVA, B. et al., 2021), e possuía 357 oportunidades estando 260 delas catalogadas em elegíveis ou não, conforme Figura 5.1a. Os dados foram obtidos através da aplicação de técnicas de *web scraping* em mais de trinta plataformas (websites) diferentes. O comprimento médio dos textos era de 800 tokens, porém com uma alta variação, podendo chegar a 5000 tokens. A distribuição do tamanho dos textos das oportunidades pode ser observada na Figura 5.1b.




(a) (b)

Figura 5.1 – Detalhamento do *dataset* Preliminar - (a) Quantidade de oportunidades elegíveis, não elegíveis e não catalogadas; (b) Tamanho dos textos em número de palavras (sem pré-processamento).

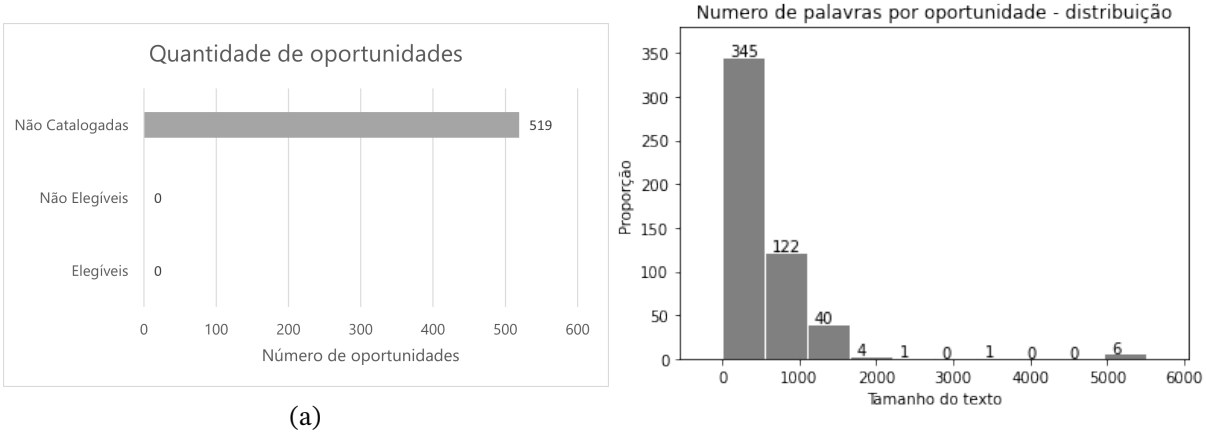
## 5.2 Dataset PPF

O *dataset* PPF foi solicitado ao grupo de raspagem e corresponde a uma prévia do *dataset* Final com oportunidades que já haviam sido raspadas em iterações anteriores dos scripts de raspagem, mas que ainda não foram devidamente rotuladas. Os textos dessas oportunidades, apesar de não catalogados, podem ser utilizados para um pré-treinamento de maneira não-supervisionada.

Ele consiste em 519 oportunidades, todas não catalogadas (Figura 5.2a). A distribuição do comprimento dos textos do *dataset* pode ser observado na Figura 5.2b onde pode-se observar que parte considerável delas possuem tamanhos superiores a 500 palavras e, mesmo possuindo mais oportunidades que o *dataset* preliminar, poucas ultrapassam 2000 palavras.

## 5.3 Dataset Final

O *dataset* Final foi a entrega realizada pela equipe de raspagem referente à meta 4 (ver Tabela 2.1, contendo 928 oportunidades raspadas de diferentes instituições. Algumas




(a) (b)

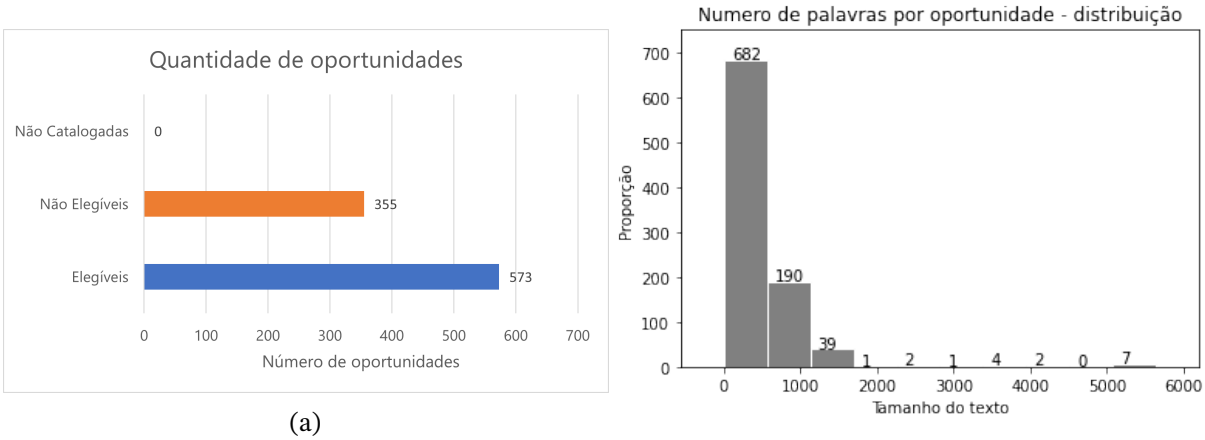
Figura 5.2 – Detalhamento do *dataset* PPF - (a) Quantidade de oportunidades elegíveis, não elegíveis e não catalogadas; (b) Tamanho dos textos em número de palavras (sem pré-processamento).

destas instituições estão listadas a seguir:

- Deutsche Forschungsgemeinschaft
- Foundation for Food & Agriculture Research
- International Centre for Genetic Engineering and Biotechnology
- Netherlands Enterprise Agency
- Roddenberry Foundation
- Scherman Foundation
- The Academy of Medical Sciences
- The Royal Society
- The Waterloo Foundation
- The World Academy of Sciences

Em comparação com o *dataset* PPF, o *dataset* Final conta com 682 novas oportunidades adquiridas durante a etapa final de raspagem, considerando somente aquelas adquiridas com os scripts de raspagem mais atualizados, uma vez que alterações na estrutura das plataformas pode inviabilizar o uso de alguns scripts.

No contexto do projeto isso é importante porque significa mais dados únicos de treino e validação para a rede. Essas oportunidades foram devidamente catalogadas pelo próprio time do projeto em conjunto com a equipe do MCTI e serviram como base para o treinamento da rede de classificação.




(a) (b)

Figura 5.3 – Detalhamento do *dataset* Final - (a) Quantidade de oportunidades elegíveis, não elegíveis e não catalogadas; (b) Tamanho dos textos em número de palavras (sem pré-processamento).

Como é possível observar na Figura 5.3a, o *dataset* foi totalmente catalogado, apresentando 573 oportunidades elegíveis e 355 não elegíveis, com um tamanho médio textual de 450 palavras. A distribuição do número de palavras por oportunidade pode ser observado na Figura 5.3b.

# 6 Metodologia e Implementação

Com os objetivos e fundamentos claros, é importante então traçar uma trajetória até os resultados. Após a revisão bibliográfica, ficou evidente que existem vários caminhos e modelos que podem ser capazes de realizar a classificação de maneira satisfatória, cada um com seus prós e contras.

Abordar a questão através do uso das técnicas mais avançadas da literatura pode ser superdimensionar o problema, com uma solução que possivelmente demandaria, além do esforço de implementação, recursos computacionais desnecessários.

No contexto do projeto, foi feito uso do Google Colab Pro como recurso computacional para o treinamento dos modelos. O ambiente utiliza a placa de vídeo Tesla P100 com 16GB disponíveis de memória de vídeo e 25GB de memória RAM para uso com o CPU no "ambiente de alta RAM". Este ambiente com RAM elevada possui um limite mensal de horas, então precisou ser utilizado com ponderação.

Decidiu-se então por uma abordagem incremental, realizando testes com arquiteturas e estratégias mais simples e, após a análise dos resultados obtidos em cada etapa, sugerir melhorias de acordo com o diagnóstico considerado.

Neste capítulo será realizada a descrição da modelagem e implementação das soluções utilizadas em todas as etapas do projeto, destacando as percepções obtidas e as dificuldades encontradas.

## 6.1 Modelagem das Soluções: Representações Textuais

Para aplicação das técnicas de Aprendizagem Profunda, primeiramente é necessário transformar o texto em informações numéricas (representação vetorial) que contextualizam o emprego de uma palavra (conjunto de caracteres) e sua relação com as demais palavras, para que, por fim, estas informações possam ser introduzidas na rede neural modelada. Para isso, foram utilizados *Word Embeddings* e *Document-Long Embeddings*, mostradas na Figura 6.1.

Na abordagem incremental proposta, optou-se por primeiro utilizar *embeddings* treinados juntos com a rede neural total, através da inclusão da camada *Embedding* do *Keras*, citado na Subseção 4.7.1. Adiante, o próximo incremento adotado foi de utilizar *Word Embeddings* pré-treinados através da técnica *Word2Vec*. Por fim, o último passo incremental foi utilizar um *Transformer* para construir representações do texto completo (ao invés de

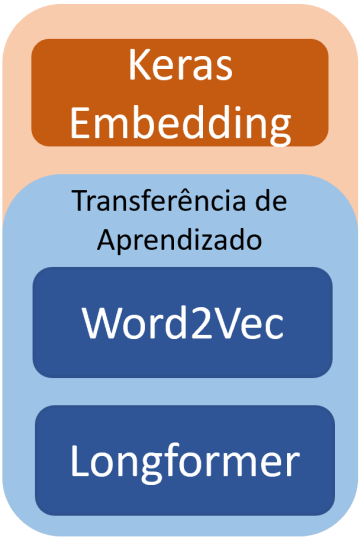



Figura 6.1 – Técnicas de representação vetorial utilizadas. Adaptado de (ROCHA. et al., 2022).

palavras ou frases) fazendo o uso do mecanismo de atenção. Levando-se em consideração os objetivos do projeto, o *Transformer* escolhido foi o modelo *Longformer*.

A Tabela 6.1 apresenta um resumo compararativo das técnicas de representação vetorial utilizadas.

Tabela 6.1 – Comparativo das técnicas de representação vetorial utilizadas.

| | ***Keras Embedding*** | ***Word2Vec*** | ***Longformer*** |
|---|---|---|---|
| **Dimensionalidade** | 8 dimensões / token | 300 dimensões / token | 768 dimensões / token |
| **Estratégia de treinamento** | Treinado com a rede | Pré-treinado com dados não rotulados | Base pré-treinada (*longformer-base-4096*) |
| **Custo computacional** | Baixo | Médio | Alto |

### 6.1.1 Keras Embedding + Deep Learning

A primeira abordagem é a mais simples, e envolve apenas a utilização de uma camada de *Embedding* cujo tamanho pode ser definido. Para o projeto, pensando em manter o custo computacional baixo para a primeira etapa, escolheu-se 8 dimensões para representar os *embeddings*.

Como a camada é acoplada à rede neural, ela aprende junto com o treinamento supervisionado da rede completa, portanto, necessita de dados catalogados para ter seus

pesos definidos e ajustáveis.

Esta é a única das técnicas abordadas que não utiliza transferência de aprendizado. Mediante a revisão bibliográfica realizada, foi possível perceber que a maior parte das arquiteturas utilizadas para classificação de texto possuem uma grande quantidade de dados para a realização do treinamento supervisionado. Assim, considerando a pequena quantidade de dados catalogados disponível para o treino, é possível presumir a necessidade de algum tipo de transferência de aprendizado para obter os melhores resultados possíveis.

Apesar de ser a implementação mais simples, acredita-se que seu resultado seja superior à técnica de BOW devido aos *Word Embeddings* serem atualizados com cada época de treinamento, ou seja, a cada ciclo em que todos os dados passaram pelo modelo, permitindo que os pesos se adequem melhor para a solução proposta.

### 6.1.2 Word2Vec + Deep Learning

A segunda abordagem faz uso de *Word2Vec Embeddings* para realizar a representação das palavras. O treinamento deste modelo como já explicado na Subseção 4.3.4.1 é realizado de maneira não-supervisionada, portanto é possível treinar os *Word Embeddings* com dados não-catalogados. Dessa forma, esses *embeddings* podem ser mais bem treinados e, com isso, melhorar o desempenho da classificação em relação aos *Keras Embeddings* da etapa anterior.

Para que este pré-treinamento, mostrado na Figura 6.2, resulte em melhores representações será necessário a utilização de um *dataset* mais robusto que possua um alto nível de similaridade com o tipo de texto a ser trabalhado no modelo. Com isso em mente foi utilizada a Equação 6.1 para calcular a similaridade entre *datasets*, que foi utilizada como critério comparativo entre diferentes *datasets* considerados.

$$P_C = \frac{\#(\{Base\} \cap \{New\})}{\#\{Base\}} * 100 \tag{6.1}$$

onde *{Base}* é o conjunto de palavras únicas do *dataset* do MCTI, *{New}* é o conjunto de palavras únicas do *dataset* alvo, e # denota o número de elementos de um conjunto. Esta fórmula calcula a porcentagem das palavras no *dataset* original presente no novo *dataset*. O número máximo é 100 (quando todos os *tokens* estão presentes no novo conjunto de dados).

Outro detalhe importante para o treinamento do modelo é a quantidade de dimensões. Por padrão o *Word2Vec* utiliza 300 dimensões e esse valor foi respeitado neste trabalho para que seja possível aproveitar o modelo da maneira desenhada na Figura 4.1.

O modelo pré-treinado servirá como um dicionário, que traduzirá as palavras em suas representações vetoriais que servirão como entrada para a rede neural profunda de classificação. Outra opção seria acoplar o modelo na rede e realizar o ajuste fino *(fine tuning)* dos pesos, porém esse processo utiliza mais recursos computacionais.

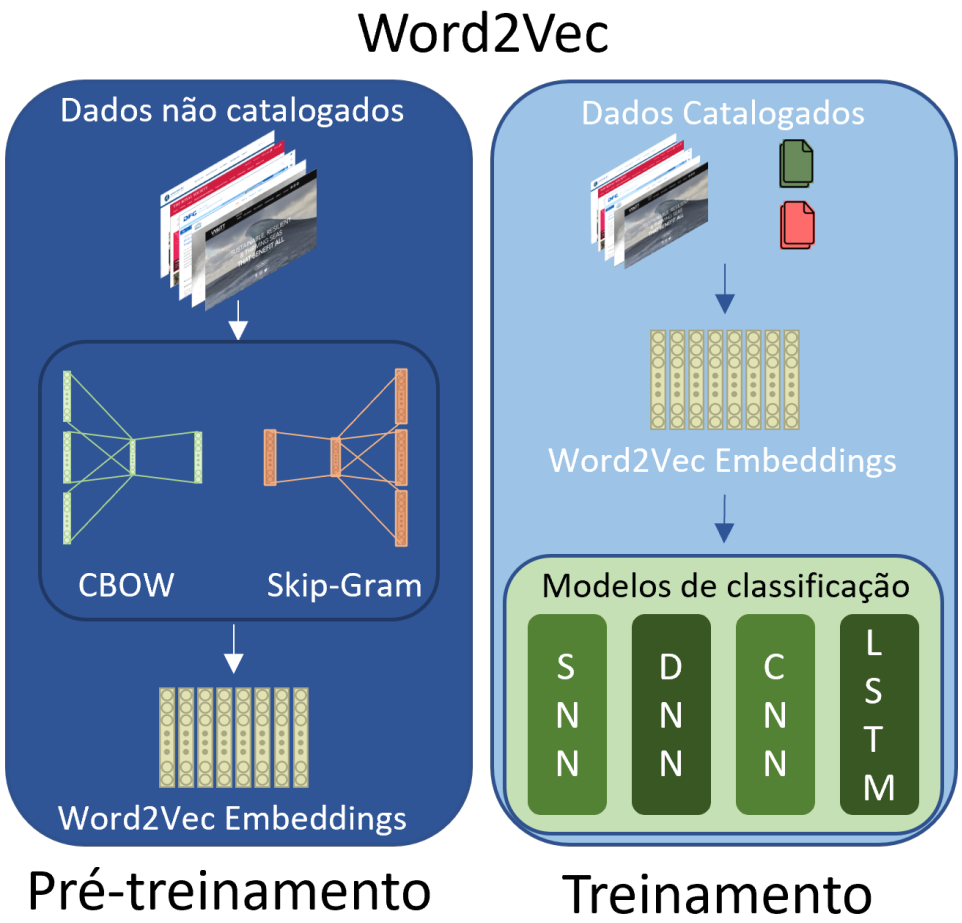



Figura 6.2 – Treinamento do modelo *Word2Vec*.

### 6.1.3 Longformer + Deep learning

A terceira abordagem utilizada para as representações vetoriais foi o *Longformer* (BELTAGY et al., 2020). Ele foi escolhido por causa da limitação da primeira geração de *Transformers* e arquiteturas baseadas em BERT envolvendo o tamanho das sentenças: o máximo de 512 *tokens*.

O *Longformer* permitiu o processamento de sequências de até 4096 caracteres sem enfrentar o gargalo de memória das arquiteturas do tipo BERT e alcançou estado da arte em diversos *benchmarks*.

Apesar de o *Longformer* ter otimizado o uso da memória em relação aos modelos baseados em BERT ele ainda necessita de grande poder computacional para ser utilizado. Nos experimentos de (BELTAGY et al., 2020) foram utilizadas placas de GPU RTX8000 com 48GB de RAM GDDR6, de poder computacional superior à GPU Tesla P100 disponível no Colab Pro, que possui apenas 16GB de RAM.

Dada a limitação de poder computacional, para esse trabalho não foi realizado o treinamento do modelo *Lonformer*. Para aplicar o *Longformer*, foi utilizada a base pré-treinada

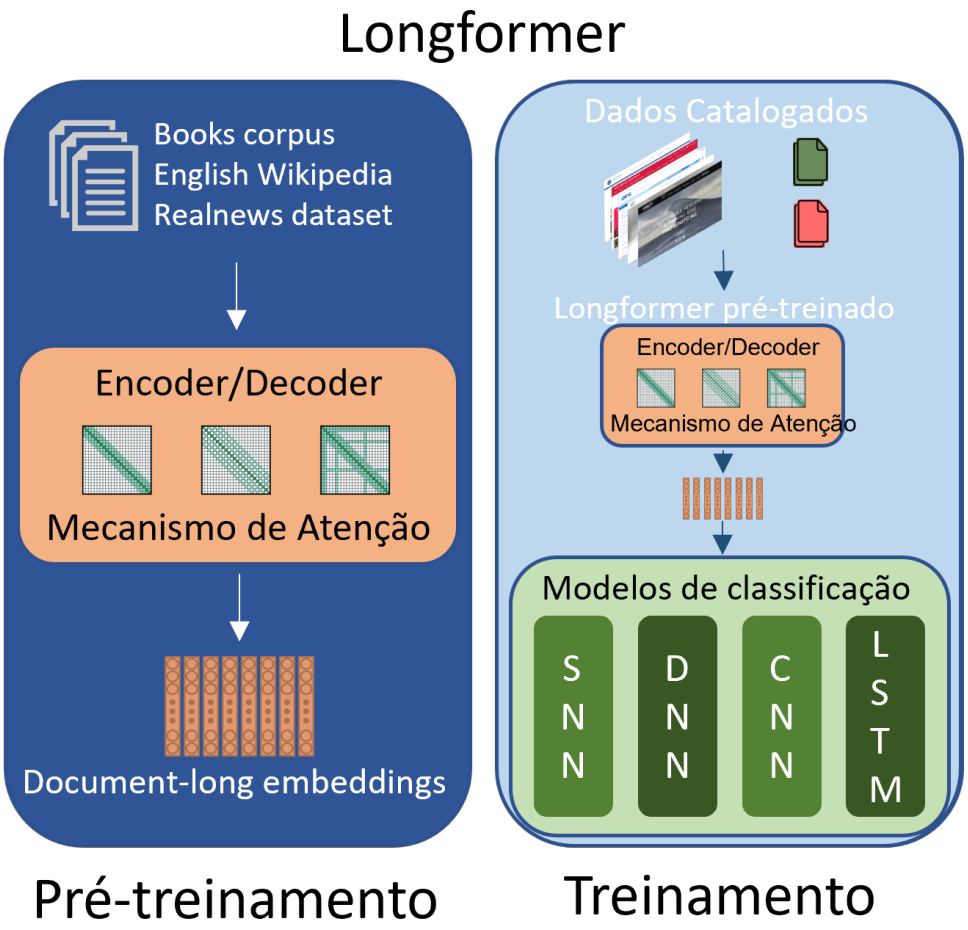



Figura 6.3 – Modelo de pré-treinamento e treinamento com o *Longformer*.

(disponível no link [1]) previamente treinada com uma combinação de vastos conjuntos de dados - *Book Corpus* (ZHU et al., 2015) mais Wikipédia em inglês e *Realnews Dataset* (ZELLERS et al., 2019) - como entrada para o modelo, conforme ilustrado na Figura 6.3.

Diferentemente das técnicas anteriores, o *Longformer* realiza *document-long embeddings*, ou seja, trabalha com o texto completo para definir sua representação vetorial. Neste caso os textos foram convertidos para suas representações vetoriais contextualizadas e então serviram de entrada para as redes neurais profundas de classificação.

Apesar de não ser possível treinar o modelo do zero, espera-se que o resultado do modelo seja superior aos outros por ser o estado da arte em classificação de textos longos, justamente pelo fato de conseguir encorporar o texto por completo, utilizando o mecanismo de atenção, para sua representação vetorial.

[1] https://huggingface.co/allenai/longformer-base-4096

## 6.2 Modelagem das Soluções: Rede Neural Acoplada

Após a representação vetorial dos textos, têm-se posse de vetores carregados de informação que podem ser utilizados como entrada para uma rede neural de classificação, a fim de analisar o texto e classificá-lo quanto a sua elegibilidade para o financiamento de pesquisas brasileiras. Dentre as métricas apresentadas na Seção 4.8, o foco principal da pesquisa foi a identificação dos métodos de classificação que apresentam a melhor acurácia.

Existem diversos tipos de arquitetura de redes neurais, com diferentes desempenhos, tempo de treinamento e consumo de recursos computacionais. Mediante as análises das principais arquiteturas presentes na literatura e considerando o escopo do projeto, as redes neurais estudadas foram *Shallow Neural Network* (SNN), *Deep Neural Network* (DNN), *Long Short-Term Memory* (LSTM) e *Convolutional Neural Network* (CNN), com a finalidade de analisar empiricamente cada arquitetura e determinar qual melhor se adequa aos objetivos do projeto.

A Figura 6.4 ilustra a estrutura e os modelos de classificação analisados e a Tabela 6.2 apresenta um resumo compararativo entre eles.

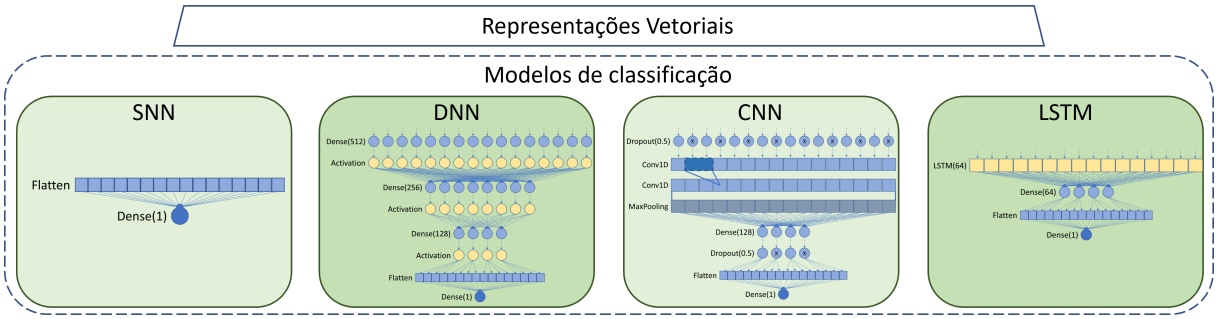



Figura 6.4 – Modelos de classificação.

Tabela 6.2 – Comparativo modelos de classificação analisados.

| | **SNN** | **DNN** | **CNN** | **LSTM** |
|---|---|---|---|---|
| **Camadas** | Flatten e Dense | Dense, Activation, Flatten | Dropout, Conv1D, MaxPooling, Dense e Flatten | LSTM, Dense e Flatten |
| **Funções de ativação** | Sigmoide | Sigmoide e ReLU | Sigmoide e ReLU | Sigmoide e ReLU |
| **Custo computacional** | Baixo | Médio | Médio | Alto |

### 6.2.1 SNN

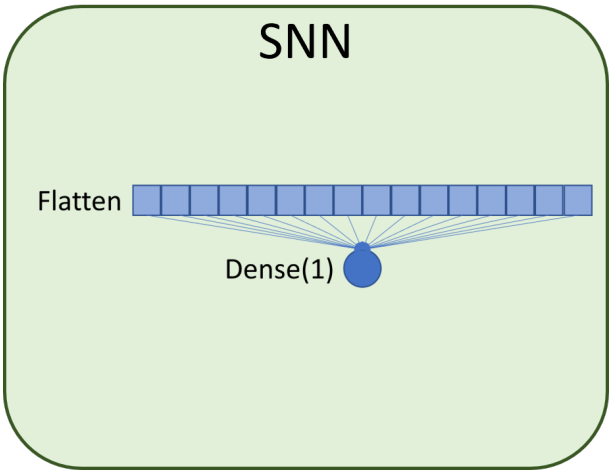



Figura 6.5 – Arquitetura da rede SNN.

Iniciando a análise pela rede neural mais simples, a SNN, seu modelo é constituído por uma camada *Flatten* simples e uma camada de classificação densa, como visto anteriormente na Subseção 4.4.4.2 e ilustrado na Figura 6.5.

O principal objetivo dessa rede é a validação das etapas de pré-processamento, formatação e validação dos dados, em uma rede neural sequencial utilizando *Keras/TensorFlow* (Seção 4.7). Como a arquitetura da SNN é extremamente simples, é possível então isolar os sistemas anteriores a rede de classificação sem necessariamente acoplá-los em uma rede mais complexa com mais pontos possíveis de falha na implementação. Além disso, pela sua simplicidade, ela é capaz de ser treinada rapidamente e com pouco custo computacional.

### 6.2.2 DNN

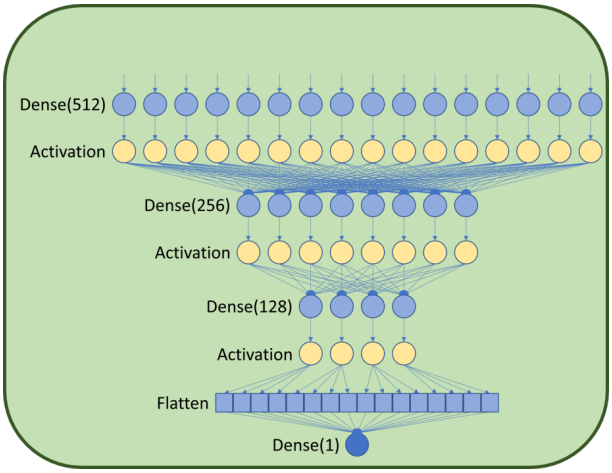



Figura 6.6 – Modelo DNN.

Incrementando a análise da solução com uma rede neural mais elaborada, a DNN, representada na Figura 6.6, consiste em um MLP (Subseção 4.4.4.1) com múltiplas camadas

densas e de ativação, fornecendo maior quantidade de parâmetros para o aprendizado.

O arquitetura modelada possui alternância entre camadas densas e de ativação ReLU, variando suas dimensionalidades de 512 a 256, até afunilar para 128, e, por fim, uma camada *Flatten* é acoplada para alimentar a camada densa de classificação final.

### 6.2.3 CNN

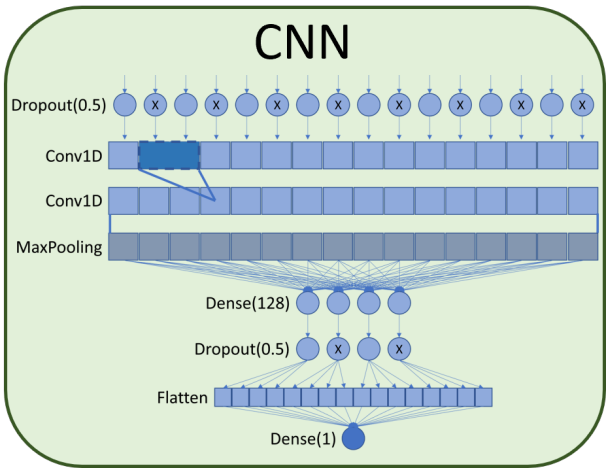



Figura 6.7 – Arquitetura da rede CNN.

A arquitetura da rede CNN utilizada é composta por uma camada de *dropout* de 50%, seguida por duas camadas de convolução 1D associadas a uma camada de *MaxPooling*, conforme ilustra a Figura 6.7. Após o *MaxPooling* foi adicionado uma camada densa de tamanho 128 conectada a um *dropout* de 50%, que por fim se conecta a uma camada *Flatten* e a camada densa de classificação final. As camadas de *dropout* ajudam a evitar o *overfitting* da rede, mascarando parte dos dados para que a rede aprenda a criar redundâncias na análise das entradas.

### 6.2.4 LSTM

O modelo LSTM utilizado, representado pela Figura 6.8, consiste em uma simples camada LSTM de dimensão 64, conectada a uma camada densa também com a mesma dimensão. Após isso, adicionou-se uma camada *Flatten* e a camada densa de classificação final.

## 6.3 Análise Preliminar

No início do projeto, os dados disponíveis para a classificação consistiam em uma planilha com 357 oportunidades que, conforme apresentado na Subseção 5.1, possuía apenas 260 delas catalogadas em elegíveis ou não. Então, no primeiro momento esse *dataset* foi o alvo

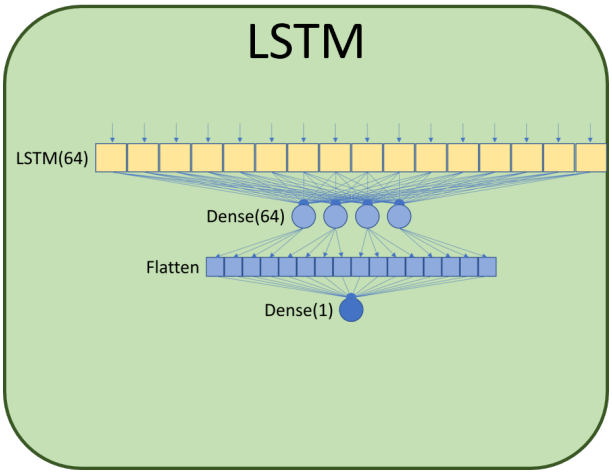



Figura 6.8 – Arquitetura de rede LSTM.

de ensaios para entender que modelos de classificação seriam eficientes para a classificação desse tipo de dados (textos longos, não estruturados e não uniformes).

O trabalho realizado para este *dataset* Preliminar serviu de base para todas as implementações seguintes, e seus resultados inspiraram a publicação do Artigo (ROCHA. et al., 2022).

Em cima desses dados foi realizada uma análise cruzada, onde todas as técnicas de representação vetorial descritas neste capítulo foram testadas em conjunto com os modelos de classificação, também apresentados neste capítulo. Objetivando preparar os textos para servirem de entrada para os modelos, foi realizado um pré-processamento básico que consistiu apenas na remoção de caracteres especiais e a tokenização do texto.

Todos os modelos foram treinados pelo menos 20 vezes e por 100 épocas de treinamento, um número bastante elevado para aplicações de processamento de linguagem natural, porém necessário dada a necessidade de identificar o comportamento do aprendizado ao decorrer das épocas. Os resultados obtidos estão apresentados na Seção 7.1.

Para o treinamento da rede foram adicionadas algumas funções de *callback* que são executadas no final de cada época de treino. O treinamento padrão através da função *fit* do *Keras* utiliza apenas a métrica de acurácia, portanto foi necessário implementar as funções de cálculo das métricas de F1-score, Recall e Precisão (Seção 4.8) como *callbacks*. Além disso, foi definido um critério de saída que interrompe o treinamento da rede caso não haja melhora de acurácia após a passagem de 20 épocas de treinamento. Esse valor foi ajustado após a análise das curvas de treinamento, apresentada na Seção 7.1.

Inicialmente foi modelada a solução utilizando *Word Embeddings* através da camada de *Embedding* do *Keras*, que treina seus pesos em conjunto com a rede neural de classificação. O treinamento nesta primeira etapa foi bastante veloz, considerando a pequena quantidade de dados, simplicidade do modelo e a utilização de um ambiente de GPU para os treinos. Os

únicos modelos que levaram acima de 1s por época de treino foram o CNN e o LSTM. Devido à pouca quantidade de dados catalogados disponíveis, entendeu-se que seria útil fazer uso de técnicas de transferência de aprendizado para melhorar o desempenho da classificação no geral.

A utilização do *Word2Vec* permitiu fazer o uso de todos os 357 dados, mesmo que não catalogados, para o treinamento não supervisionado dos pesos. Porém, como já comentado na Subseção 6.1.2, percebeu-se a necessidade de treinar os pesos *Word2Vec* com mais dados e foi realizada uma análise exploratória para a identificação de *datasets* com similaridade de palavras através da utilização da Equação 6.1.

Foram analisados diversos *datasets* encontrados através da plataforma *Kaggle*, que possui milhares de *datasets* para processamento de linguagem natural disponíveis, porém não foi possível encontrar na literatura dados adequados. Os resultados na análise exploratória podem ser observados na Tabela 6.3. O *dataset* com maior nível de compatibilidade encontrado foi o *BBC News Articles dataset* que possuía compatibilidade de 57% ou seja pouco mais da metade das palavras estavam presentes.

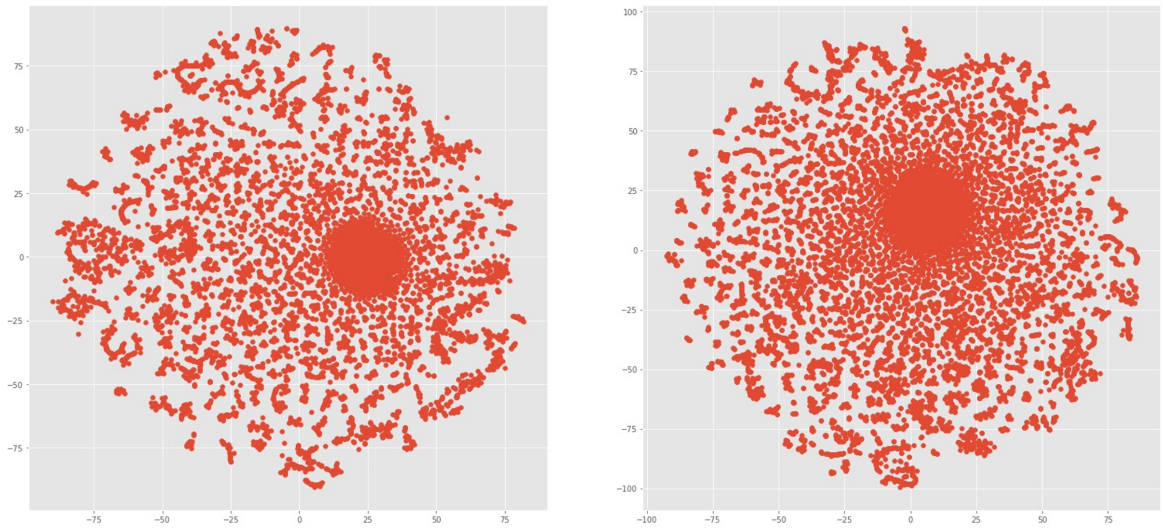
Tabela 6.3 – Resultados de compatibilidade com a base (Traduzido de (ROCHA. et al., 2022)).

| Dataset | Compatibilidade com a base |
|---|---|
| Dataset Preliminar (Catalogado) | 100% |
| Dataset Preliminar (Total) | 100% |
| BBC News Articles | 56,77% |
| Dataset PPF | 75,26% |

Por esta razão, foi solicitado ao grupo de raspagem uma prévia dos dados mais recentes, mesmo que ainda não catalogados (*dataset* PPF). A análise de compatibilidade resultou em um valor acima de 75% para o PPF em relação ao *dataset* Preliminar, reforçando a suposição de que os dados de oportunidades utilizam palavras similares.

Com a adição do *dataset* PPF no pré-treino dos pesos *Word2Vec* foi possível perceber um ganho de representação e contexto que pode ser observado na Figura 6.9.

A Figura 6.9 mostra uma representação dos pesos treinados pelo *Word2Vec*. Ela foi obtida através da técnica *t-Distributed Stochastic Neighbor Embedding (t-SNE)* (MAATEN; HINTON, 2008) que realiza redução de dimensionalidade. Em 6.10a o treinamento foi feito utilizando os 260 dados catalogados; em 6.10b foram utilizados os 876 dados obtidos da junção do *dataset* Preliminar e do *dataset* PPF. Na imagem é possível observar que a área concentrada (densa) aumentou consideravelmente com a adição do novo *dataset* e seus tokens, demonstrando o enriquecimento dos pesos no que diz respeito a informações contextuais. Se o pré-treino fosse realizado em um *dataset* incompatível, provavelmente outras áreas concentradas apareceriam e possivelmente interfeririam no significado das palavras desejadas.



(a) (b)

Figura 6.9 – Pesos *Word2Vec* pré-treinados: a) pesos treinados com o *dataset* Preliminar (Catalogado); b) pesos treinados com *dataset* Preliminar (Todos) + *dataset* PPF.

Esses *embeddings* foram pré-treinados em ambiente de GPU utilizando 1000 épocas de treinamento e esse processo demorou cerca de 2 horas utilizando os recursos do Google Colab Pro. Conforme comentado na Subseção 6.1.2, eles serviram como um dicionário, traduzindo as palavras do texto para suas representações vetoriais para que possam ser introduzidas nas redes neurais de classificação.

Como neste momento a entrada passa a possuir 300 dimensões para cada palavra, o processo de treinamento se tornou um pouco mais lento, porém a diferença para o treinamento supervisionado não foi tão grande devido a pouca quantidade de dados catalogados utilizada.

Por último foi aplicado o modelo *Longformer* para a codificação do texto em sua representação vetorial. Para o *dataset* Preliminar, sua vantagem em relação às soluções baseadas em BERT é nítida já que devido ao tamanho limitado de 512 tokens, 99 oportunidades precisariam ser truncadas podendo perder informações críticas para a classificação correta. Já com o comprimento de 4096 tokens do *Longformer*, apenas 8 oportunidades tiveram suas informações encurtadas. Conforme mencionado na Subseção 6.1.3, o modelo utilizado já havia sido pré-treinado, portando o único processamento necessário foi o de realizar a tradução do texto para suas representações vetoriais contextualizadas, para que possam ser introduzidos nas redes neurais de classificação. Esse processo em conjunto com o treinamento das redes foi bastante custoso para o computador, alcançando o limite de 16GB de RAM da placa de vídeo após apenas 3 rodadas de treinamento.

Para solucionar este problema foi necessária a redução do tamanho do *batch* de treinamento, ou seja, do número de exemplos de treinamento utilizados em uma iteração, além da realização do processo de desalocação da memória após cada etapa.

Apesar do grande tamanho do modelo carregado em memória, o treinamento dos modelos não sofreu um aumento significativo no tempo de treino de cada época e pôde ser realizado sem outras complicações.

## 6.4 Pré-processamento

Na etapa detalhada na Seção 6.3, o único pré-processamento realizado foi o de remover caracteres especiais e tokenização. Com os resultados obtidos, apresentados na Seção 6.1, identificou-se a necessidade de uma maior quantidade de dados e também um melhor tratamento desses dados. Desse modo, percebeu-se que a realização de um melhor pré-processamento poderia melhorar não só a velocidade de treinamento dos modelos, mas também sua performance.

Com a disponibilização do *dataset* Final, uma vez que a quantidade de dados para o treinamento aumentou em mais de três vezes, tornou-se necessário:

- Uniformizar os textos para a língua inglesa;
- Reduzir o número de tokens/palavras insignificantes para a tarefa proposta;
- Tornar o texto mais limpo para otimizar o treinamento dos modelos.

A planilha com os novos dados originalmente possuía algumas oportunidades com textos em português que necessitavam de tradução para o inglês, para que fosse possível serem interpretados pela mesma rede. Essa tradução foi realizada através da biblioteca de python *translators* (ULIONTSE, 2023) utilizando o tradutor do Google. Devido ao limite apresentado pela ferramenta de 5000 caracteres para a entrada de texto, foi necessário quebrar os textos em frases para a realização das traduções.

Nesta fase de tradução também foi encontrado um problema de inconsistência nos resultados da tradução, que variava um pouco a cada rodada de tradução. Essa variação só foi percebida quando houve a necessidade de replicar os resultados, o que não foi possível devido à tradução não corresponder exatamente aos textos utilizados previamente. Devido a esse problema foi necessário realizar a tradução novamente para atualizar o *dataset* e remover a etapa de tradução do conjunto de operações de pré-processamento.

Outro tratamento realizado foi o de consolidação de algumas informações disponíveis nas páginas em apenas uma única sequência textual que pode ser trabalhada como entrada para rede.

A partir dos dados raspados e catalogados no projeto, pode-se perceber uma divergência em alguns dos dados contidos na coluna opo_texto e opo_texto_ele. Para realizar o

trabalho de classificação será necessário utilizar apenas uma base de texto, e para isso foram definidas as seguintes regras:

1. Se: opo_texto IGUAL opo_texto_ele => Utiliza-se opo_texto;
2. Se: opo_texto DIFERENTE opo_texto_ele E opo_texto_ele IGUAL "nan"=> Utiliza-se opo_texto;
3. Se: opo_texto DIFERENTE opo_texto_ele E opo_texto_ele DIFERENTE "nan" E n_tokens(opo_texto) < 4000 => Utiliza-se opo_texto + opo_texto_ele;
4. Outros casos => Utiliza-se opo_texto

Utilizando essas regras dos 928 dados de treino:

- 795 obedecem a regra 1
- 18 obedecem a regra 2
- 114 obedecem a regra 3
- 1 obedece a regra 4

Após esses dois tratamentos foram adicionadas técnicas de pré-processamento. Existem diversas técnicas que podem ser utilizadas para reduzir a quantidade de palavras únicas em um texto sem necessariamente afetar seu entendimento e interpretação pelo computador. Para o projeto foram selecionadas as técnicas de:

- Expansão de contrações;
- Transformacão do texto para minúsculo;
- Remoção de pontuações;
- Stemização;
- Lematização;
- Remoção de stopwords.

Essas técnicas foram combinadas para compor 8 experimentos realizados de maneira a identificar a melhor abordagem, levando em consideração a redução na quantidade de tokens únicos e tamanho de sentenças, além da melhoria de performance e acurácia. Para avaliar a melhora do desempenho com os dados pré-processados, os experimentos foram acoplados à rede neural de classificação mais simples (*Keras Embedding* + SNN), que serviu

para validar que os dados pré-processados podem ser utilizados para o treino de uma rede de classificação.

Primeiro, avaliou-se o tratamento da pontuação e capitalização. Esta fase resultou na construção e avaliação dos quatro primeiros experimentos (Xp1, Xp2, Xp3, Xp4). Em seguida, avaliou-se a simplificação do conteúdo, a partir da base de dados resultante do experimento Xp4, considerando a stematização (Xp5), lematização (Xp6), stematização + remoção de stopwords (Xp7), e lematização + remoção de stopwords (Xp8).

Tabela 6.4 – Experimentos de pré processamento realizados

| **Base** | **Textos originais** |
|---|---|
| Xp1 | Expandir contrações |
| Xp2 | Expandir contrações + Transformar texto em minúsculo |
| Xp3 | Expandir contrações + Remover pontuação |
| Xp4 | Expandir contrações + Remover pontuação + Transformar texto em minúsculo |
| Xp5 | Xp4 + Stemização |
| Xp6 | Xp4 + Lematização |
| Xp7 | Xp4 + Stemização + Remoção de stopwords |
| Xp8 | Xp4 + Lematização + Remoção de stopwords |

A partir dos experimentos, descritos na Tabela 6.4, foi possível identificar as técnicas que melhor se adequam para o tipo de dado utilizado, avaliando a melhora de performance do sistema e efetividade na redução do tamanho dos textos de entrada, de maneira a garantir que o mínimo de informação seja perdido.

## 6.5 Análise Final

Na etapa de análise preliminar, Seção 6.3, foi possível identificar modelos de aprendizado profundo capazes de classificar os textos de oportunidade. Já na etapa de pré-processamento, Seção 6.4, foi possível identificar as melhores técnicas de tratamento dos dados para otimizar o desempenho do sistema. Na presente etapa, o *dataset* Final já estava disponível, então o objetivo atual é consolidar as melhores técnicas do *dataset* mais completo na tentativa de obter melhores resultados. O novo *dataset* possui 928 dados catalogados, então espera-se resultados superiores aos encontrados em (ROCHA. et al., 2022), já que a rede poderá aprender, de maneira supervisionada, através de mais exemplos.

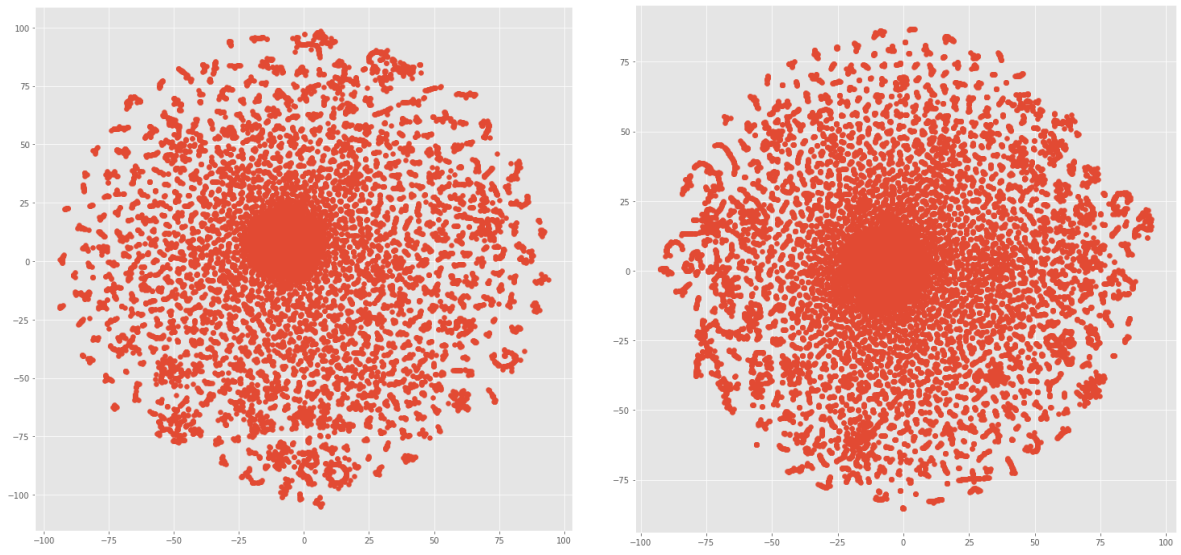
Similarmente à etapa preliminar, foi realizada uma análise cruzada com as técnicas de representação vetorial e os modelos de aprendizagem profunda para que seja possível comparar os modelos e identificar o mais adequado para a solução final.

A aplicação dos *embeddings* com a camada do *Keras* foi simples e sem complicações. Um detalhe importante a ser destacado é de que os testes com o modelo SNN foram realizados

já na etapa de pré-processamento como validação, e por isso essa rede foi treinada mais vezes, aumentando as chances de obter melhores inicializações e assim, melhores resultados.

Para os *Word2Vec Embeddings* foi realizado um pré-treino com os 928 dados do *dataset* Final. Esse pré-treino se aproximou do limite de memória de vídeo da placa disponível no Google Colab Pro, e necessitou de algumas otimizações, principalmente para liberação de memória, para que fosse possível viabilizar o treinamento. Devido aos pesos serem treinados com os mesmos dados que o *Keras Embeddings* espera-se que a diferença no resultado dos dois tipos de representação seja menor do que na análise preliminar, onde o *Word2Vec* possuiu mais dados de treino em relação ao outro modelo.

Pensando em melhorar os *Word Embeddings* deste modelo, um outro conjunto de pesos *Word2Vec* foi treinado com um *dataset* combinado com as oportunidades disponíveis nos *datasets* Preliminar, PPF e Final. A Figura 6.10, também obtida através da técnica t-SNE, mostra uma representação dos pesos treinados. Com o pré-treino utilizando esse *dataset combinado*, com mais de 1500 textos de oportunidade, houve um enriquecimento dos pesos com a adição de novas palavras dentro da área conexa, o que pode ser observado pelo aumento da densidade dos pontos próximo ao centro.



(a) (b)

Figura 6.10 – Pesos *Word2Vec* pré-treinados: a) pesos treinados com o *dataset* Final; b) pesos treinados com *dataset* combinado (Preliminar + PPF + Final).

Esses dois conjuntos de pesos pré-treinados foram utilizados para a realização do treinamento das redes, e seus resultados são apresentados e comparados na Subseção 7.3.2. Dessa forma é possível contrastar o resultado e avaliar o ganho trazido por um pré-treino com mais dados.

Para a aplicação do *Longformer*, um grande problema surgiu em relação ao tamanho do modelo. A Tabela 6.5 apresenta os tamanhos relativos aos modelos de representação textuais utilizados. Nela é possível observar que o peso do modelo *Longformer* é muito

superior ao dos outros modelos. Isso significa que carregar as representações do *Longformer* em memória necessita de 11Gb disponíveis apenas para o modelo, sem considerar o peso da rede de aprendizado profundo. Para isso, é necessário uma GPU de mais de 20 Gb para realizar o treinamento, o que não se reflete nos recursos disponíveis do projeto, que utiliza a placa Tesla P100 com 16Gb de memória.

Tabela 6.5 – Tamanhos dos modelos de representação textuais para a classificação do dataset Final.

| **Modelo de representação** | **Tamanho** |
|---|---|
| *Keras Embedding* | Tamanho incluído na rede final (< 1 Mb) |
| *Word2Vec* treinado com o *dataset* Final | 56,1 Mb |
| *Word2Vec* treinado com o *dataset* combinado (Preliminar + PPF + Final) | 85,2 Mb |
| *Longformer* | 10,88 Gb |

Devido a essa limitação, foi necessário utilizar ambiente de CPU para o treino das redes. Neste caso foi utilizado o ambiente de alta RAM do Google Colab Pro, que entrega 25GB de memória para uso com o CPU. Isso implica em maior tempo necessário para o treino, já que a GPU realiza operações matriciais mais rapidamente.

Assim como na análise Preliminar da Seção 6.3, os modelos foram treinados 20 vezes com 100 épocas de treinamento, com exceção dos modelos que utilizam o *Longformer* que, devido ao seu longo tempo de execução, foram treinados apenas 5 vezes.

## 6.6 Implantação do Protótipo

Objetivando facilitar a utilização da ferramenta de classificação pelo MCTI, foi desenvolvido um protótipo de aplicação e disponibilizado através da ferramenta *Hugging Face Spaces* (FACE, 2023). Nela, é possível subir o modelo treinado de classificação e construir uma interface para facilitar seu uso.

Para a construção da interface foi utilizada a biblioteca de python *Gradio* (GRADIO, 2023), que permite a criação de interfaces interativas com o foco em projetos de aprendizagem de máquina. A Figura 6.11 apresenta o protótipo implantado.

O protótipo implantado possui como entrada uma planilha excel com os dados das oportunidades e possui como opções de operação: **Pré-processamento + Classificação**, **Apenas Pré-processamento** e **Apenas Classificação**. Para as opções com pré-processamento a planilha precisa possuir os campos *opo_texto* e *opo_texto_ele* para permitir

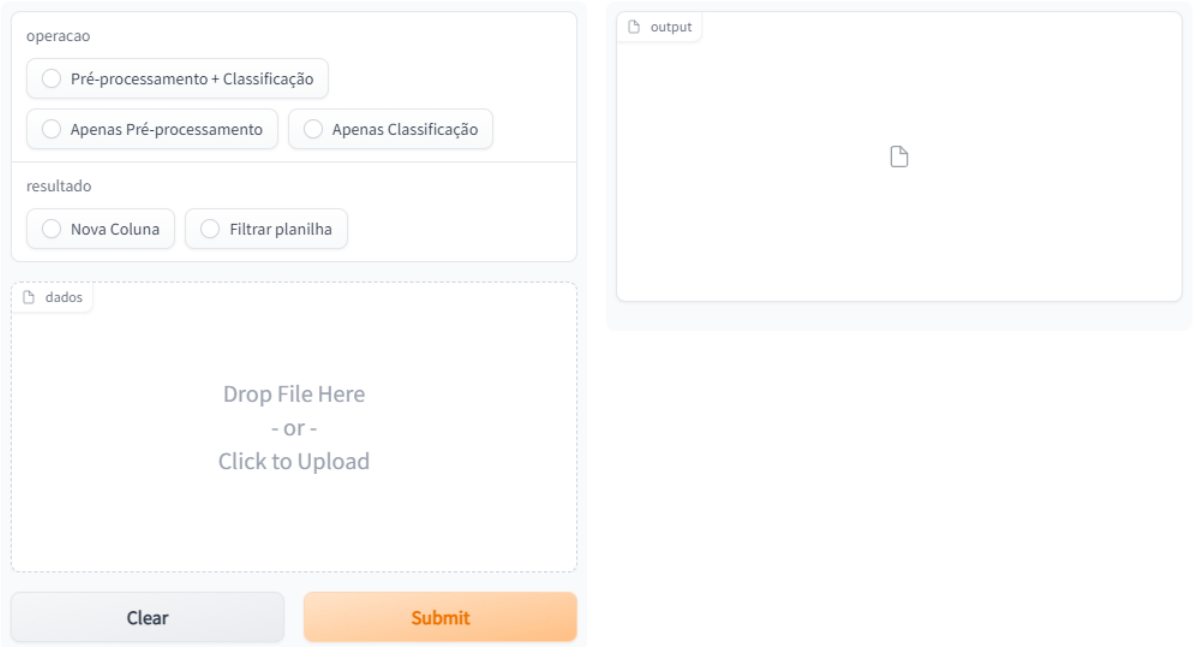



Figura 6.11 – Captura de tela do protótipo implantado.

que o sistema realize as operações e crie um novo campo de *opo_pre_tkn* com o texto já pré-processado e tokenizado.

Para a operação de classificação o sistema faz uso do campo *opo_pre_tkn* para executar o modelo e classificar as oportunidades. O tipo de resultado pode ser escolhido com as opções na interface: **Nova Coluna**, que adiciona uma coluna *classification* a mais na planilha com o valor 1 para as oportunidades elegíveis e 0 para as não elegíveis, e **Filtrar planilha**, que mantém apenas as oportunidades classificadas como elegíveis na planilha de saída.

# 7 Experimentos e Resultados

Conforme comentado no capítulo 6 foram realizados alguns experimentos para a validação e análise dos modelos e técnicas implementados: Análise Preliminar; Experimentos para o Pré-processamento; e Análise Final.

A análise preliminar (Seção 6.3) consistiu em uma série de experimentos utilizando as técnicas e modelos de representação comentadas na Seção 6.1 em conjunto com os modelos de aprendizagem profunda de classificação apresentados na Seção 6.2. O objetivo desses experimentos foi a classificação do *dataset* Preliminar, apresentando na Seção 5.1, de maneira a validar os modelos de classificação e técnicas empregadas no que diz respeito à sua capacidade de classificar os textos de oportunidade de financiamento do PPF. Vale notar que nesta etapa também foi utilizado o *dataset PPF*, descrito na Seção 5.2, para a realização do pré-treino não supervisionado dos *Word Embeddings* através da técnica *Word2Vec*.

Em seguida são apresentados os resultados dos experimentos para o pré-processamento (Seção 6.4), conforme explicado na Seção 6.4, que consistiram no emprego de técnicas de PLN para melhorar a performance do sistema de classificação . Para esta etapa foi utilizado o *dataset* Final, descrito na Seção 5.3, e o modelo de classificação SNN descrito na Subseção 6.2.1.

Finalmente foi realizada a análise final (Seção 6.5), fazendo uso do *dataset* Final (Seção 6.5) que consistiu do emprego das técnicas, validadas na Análise Preliminar, para a realização da classificação. Para esta etapa, foi utilizada a base de dados resultante das melhores técnicas obtidas pelos ensaios de pré-processamento. Esta análise final representa os resultados principais desta pesquisa.

Todos os resultados obtidos podem ser replicados utilizando os notebooks disponíveis no repositório do projeto [1]. Neste capítulo serão apresentados os resultados obtidos em cada um desses experimentos e seguido por uma breve discussão dos mesmos.

## 7.1 Resultados da Análise Preliminar

Os resultados desta etapa estão divididos de acordo com o método de representação utilizado: *Keras Embedding, Word2Vec, Longformer*. Para cada um desses métodos foram acoplados os modelos de aprendizagem profunda: SNN, DNN, CNN e LSTM descritos na Subseção 4.4.4. Para esta etapa foi utilizado o *dataset* preliminar.

Esses resultados inspiraram a escrita do artigo (ROCHA. et al., 2022). As métricas

[1] Repositório do projeto: https://github.com/chapOlin/PPF-MCTI

definidas na seção 4.8 foram utilizadas para avaliar a efetividade da classificação.

### 7.1.1 Keras Embedding + Deep Learning

Os resultados de treinamento obtidos para o método de representação *Keras Embedding*, associado aos modelos de aprendizagem profunda, podem ser conferidos nas curvas de treinamento apresentadas nas Figuras 7.1, 7.2, 7.3 e 7.4.

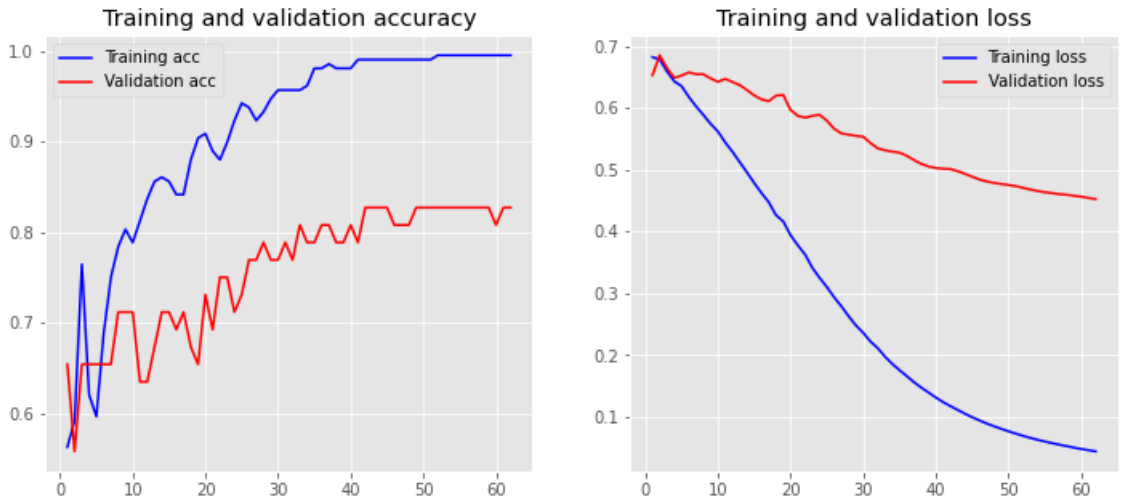



Figura 7.1 – Curvas de treinamento *Keras* + SNN.

Para o treinamento do modelo *Keras* + SNN, apresentado na Figura 7.1, foi utilizado um *callback* que interrompe o treinamento caso ele passe 20 épocas sem aprender, ou seja, sem apresentar melhorias na acurácia de validação. É possível notar também que o modelo continuou com tendência de aprendizado, o que pode ser evidenciado pelo padrão apresentado na curva da perda de validação (validation loss), que continuou decaindo. Com essa análise, percebeu-se que seria possível treinar o modelo por mais épocas, e por esse motivo, o callback foi ajustado para utilizar 50 épocas ao invés de 20 para a interrupção do treinamento. Apesar disto, o modelo foi treinado mais vezes, porém não conseguiu ultrapassar os valores de acurácia obtidos na época 42.

No modelo DNN, apresentado na Figura 7.2, o callback de interrupção foi ajustado para utilizar 50 épocas, similar ao ajuste feito no SNN. Porém o sistema apresentou grandes oscilações e obteve seus melhores resultados em torno da época 18 de treinamento. Por esse motivo, o treinamento foi interrompido antes da época 70.

Para o modelo CNN, Figura 7.3, é possível observar que a inicialização do treinamento não foi boa, pois o modelo passou algumas épocas sem aprender. Apesar disto, após a época 23 o modelo começou a aprender rapidamente e se estabilizou em torno da época 39. Dali em diante houveram pequenas oscilações, porém, próximo da época 78 houve um pequeno pico que alcançou resultado superior aos anteriores.

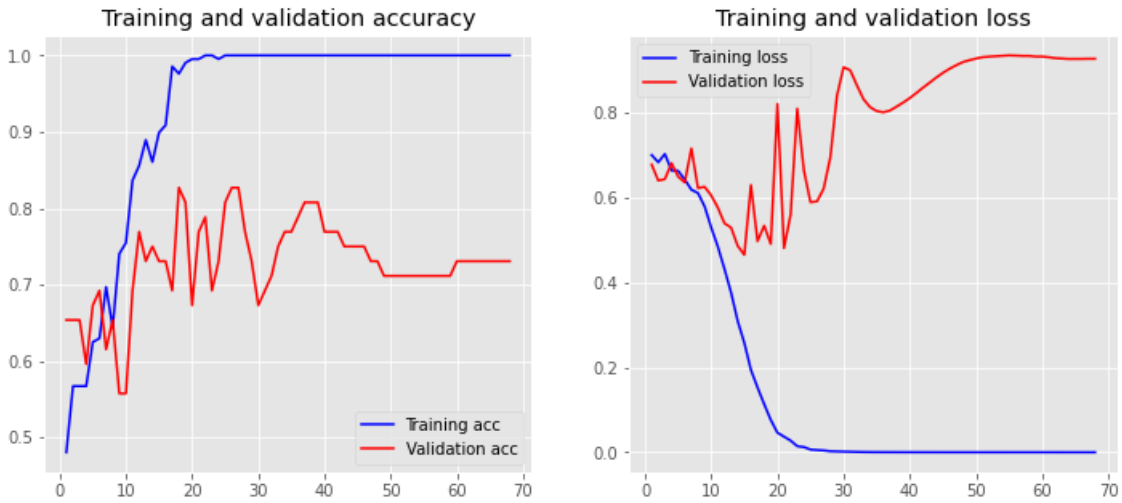



Figura 7.2 – Curvas de treinamento *Keras* + DNN.

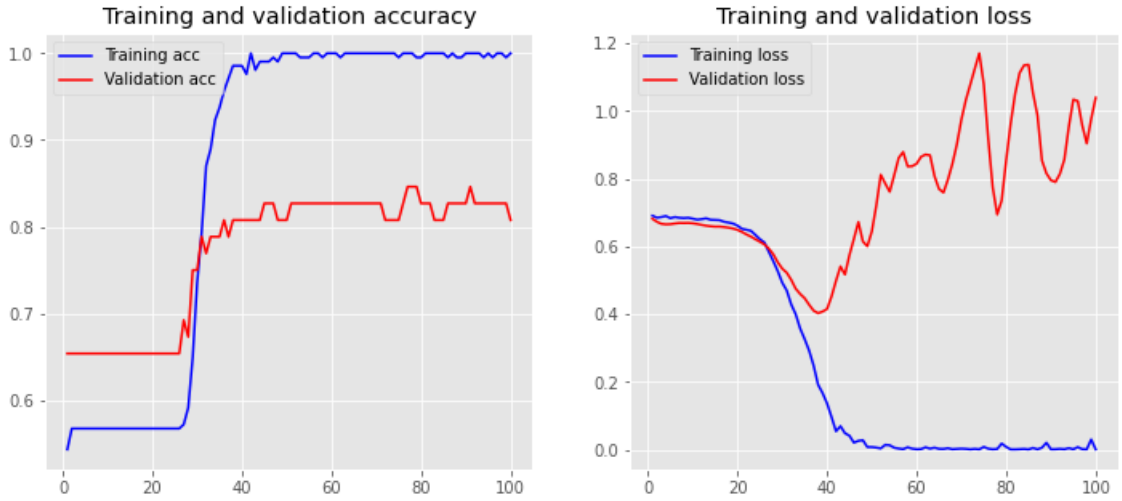



Figura 7.3 – Curvas de treinamento *Keras* + CNN.

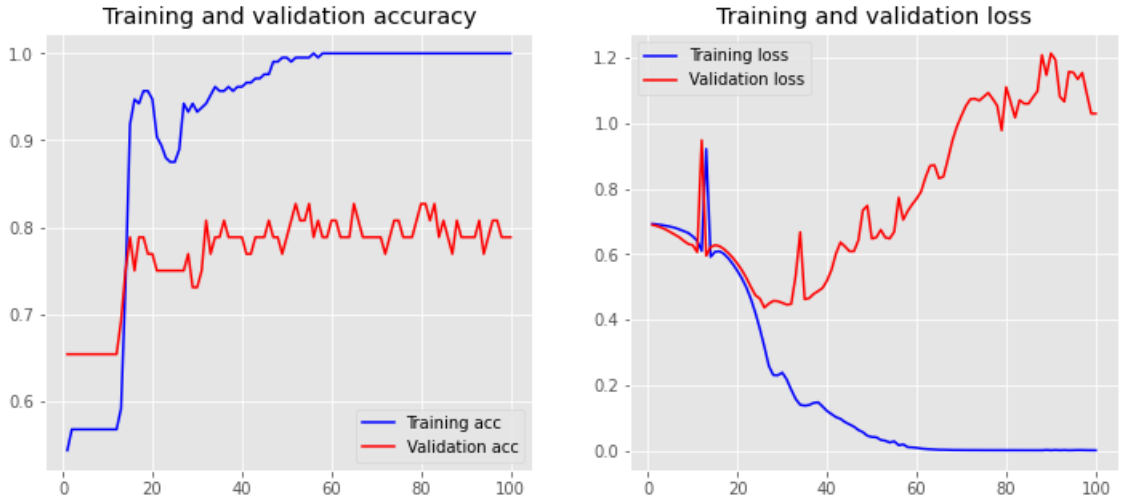



Figura 7.4 – Curvas de treinamento *Keras* + LSTM.

Assim como no CNN, o modelo LSTM (Figura 7.4) teve dificuldade para aprender no início e apresentou uma melhora substancial no seu aprendizado em torno da época 16. Além disso, é possível notar também que sua curva de acurácia mostra bastante oscilação, sendo este um padrão comum para esse tipo de modelo. Os melhores resultados de acurácia foram obtidos após a época 50.

A Tabela 7.1 apresenta um resumo dos resultados da classificação obtida para o *dataset* preliminar usando *Keras Embedding* combinado com os modelos de aprendizagem profunda.

| **Modelo** | **Acurácia** | **F1 score** | **Precisão** | **Recall** |
|---|---|---|---|---|
| SNN | 0,8269 | 0,8620 | 0,8212 | 0,9129 |
| DNN | 0,8269 | 0,8650 | 0,8952 | 0,8447 |
| CNN | 0,8462 | 0,8756 | 1,0000 | 0,7803 |
| LSTM | 0,8269 | 0,8675 | 0,8276 | 0,9129 |

Tabela 7.1 – Resultados do *Keras Embedding* combinado com os modelos de aprendizagem profunda para a classificação do *dataset* Preliminar.

Ao examinar a Tabela 7.1, é possível verificar que todos os modelos obtiveram acurácias já superiores à linha de base de 78%, atingindo valores próximos a 85% na arquitetura CNN. Outro ponto interessante a ser analisado é que esse modelo obteve uma precisão de 100%, o que significa que todas as oportunidades identificadas como elegíveis foram identificadas corretamente.

### 7.1.2 Word2Vec Pré-treinado + Deep Learning

Os resultados de treinamentos para o método *Word2Vec* associado aos modelos de aprendizagem profunda são apresentados nas curvas de treinamento contidas nas Figuras 7.5, 7.6, 7.7 e 7.8.

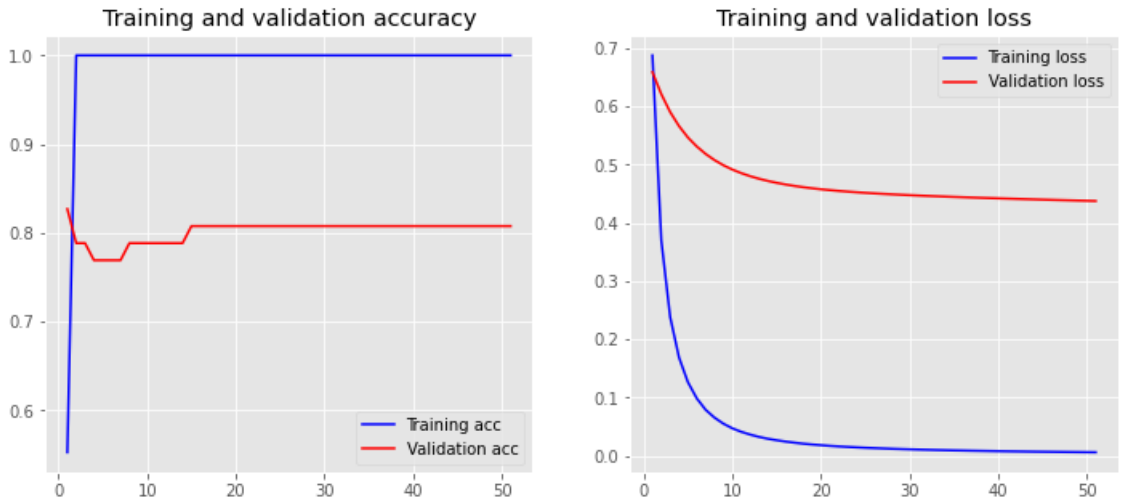



Figura 7.5 – Curvas de treinamento *Word2Vec* + SNN.

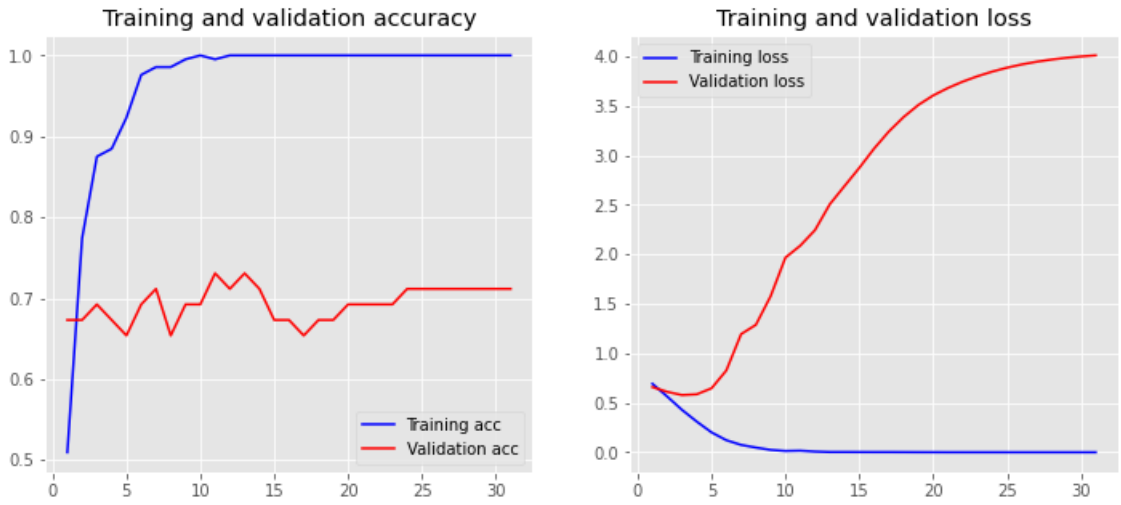



Figura 7.6 – Curvas de treinamento *Word2Vec* + DNN.

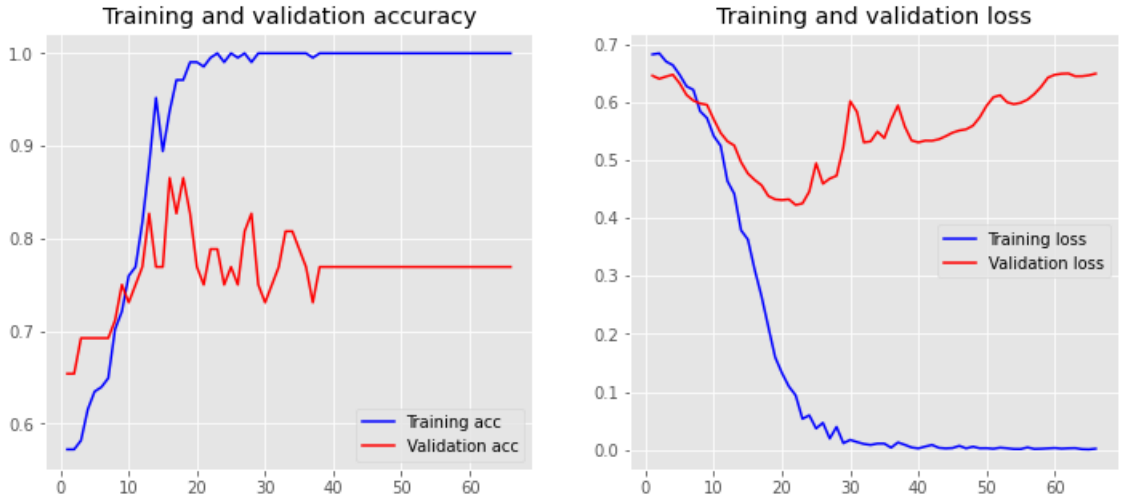



Figura 7.7 – Curvas de treinamento *Word2Vec* + CNN.

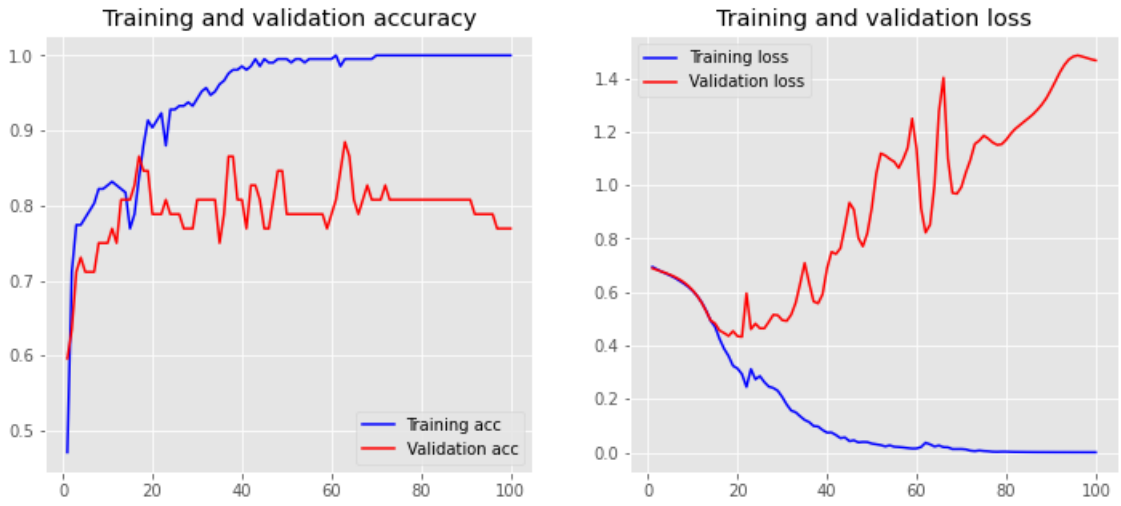



Figura 7.8 – Curvas de treinamento *Word2Vec* + LSTM.

Na Figura 7.5 é possível notar que o treinamento da SNN contou com uma excelente inicialização que não foi superada ao decorrer do treinamento, uma vez que com esse modelo a acurácia de treino alcançou 100% logo após a 2ª época de treino. Já o DNN (Figura 7.6) não contou com uma boa performance na inicialização e o modelo não conseguiu aprender corretamente, o que pode ser evidenciado pelo grande aumento da perda de validação (validation loss) logo após a 5ª época de treino. Com isso, o treinamento de épocas subsequentes foi desencorajado.

O treinamento do CNN (Figura 7.7) seguiu uma curva de aprendizado mais padrão, com o pico de acurácia em torno da época 15. Por fim, na curva de treinamento do LSTM (Figura 7.8), é possível observar as flutuações comumente presentes neste tipo de modelo para valores de acurácia. Também pode-se notar que o modelo atingiu sua melhor precisão por volta da 60ª época. Após isso, embora a acurácia de treinamento tenha melhorando, também há um padrão de *overfitting* em que a acurácia de validação declina.

A Tabela 7.2 mostra os resultados obtidos com as métricas relacionadas. Com esta implementação, foi alcançado novos níveis de acurácia de 86% para a arquitetura CNN e 88% para a arquitetura LSTM.

| **Modelo** | **Acurácia** | **F1 score** | **Precisão** | **Recall** |
|---|---|---|---|---|
| SNN | 0,8269 | 0,8545 | 0,8392 | 0,8712 |
| DNN | 0,7115 | 0,7794 | 0,7255 | 0,8485 |
| CNN | 0,8654 | 0,9083 | 0,8486 | 0,9773 |
| LSTM | 0,8846 | 0,9139 | 0,9056 | 0,9318 |

Tabela 7.2 – Resultados do *Word2Vec* combinado com os modelos de aprendizagem profunda para a classificação do *dataset* Preliminar.

### 7.1.3 Longformer + Deep Learning

As curvas de treinamento para o *Longformer*, associado aos modelos de aprendizagem profunda, são apresentadas nas Figuras 7.9, 7.10, 7.11 e 7.12.

Como é possível notar na Figura 7.9, a curva de treinamento do SNN evidencia a fragilidade do modelo, uma vez que com a quantidade limitada de parâmetros treináveis, um pequeno ajuste já foi suficiente para causar grandes mudanças na acurácia obtida, resultando em grandes oscilações para esse modelo. Por outro lado, o treinamento do DNN (Figura 7.10) atingiu seu pico de acurácia em torno da época 18 de treinamento, também apresentando grandes oscilações na curva, similarmente ao resultado da implementação com *Keras embedding* (Figura 7.2).

Já para o CNN, exposto na Figura 7.11, a melhor acurácia foi obtida próxima à 80ª época de treinamento. Também é possível observar que mesmo com a acurácia de treinamento atingindo 100%, a validação continua oscilando até atingir o valor de 84.62%.

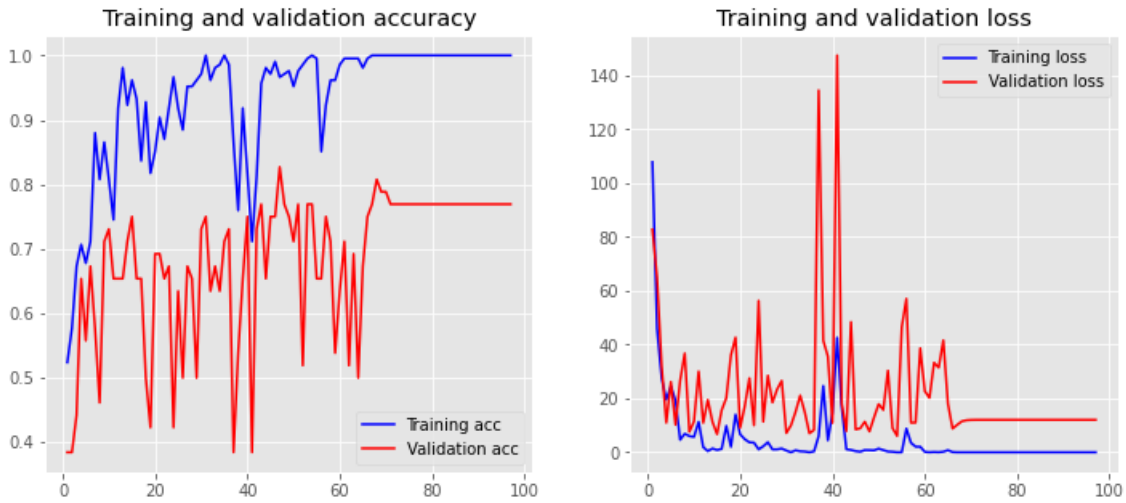



Figura 7.9 – Curvas de treinamento *Longformer* + SNN.

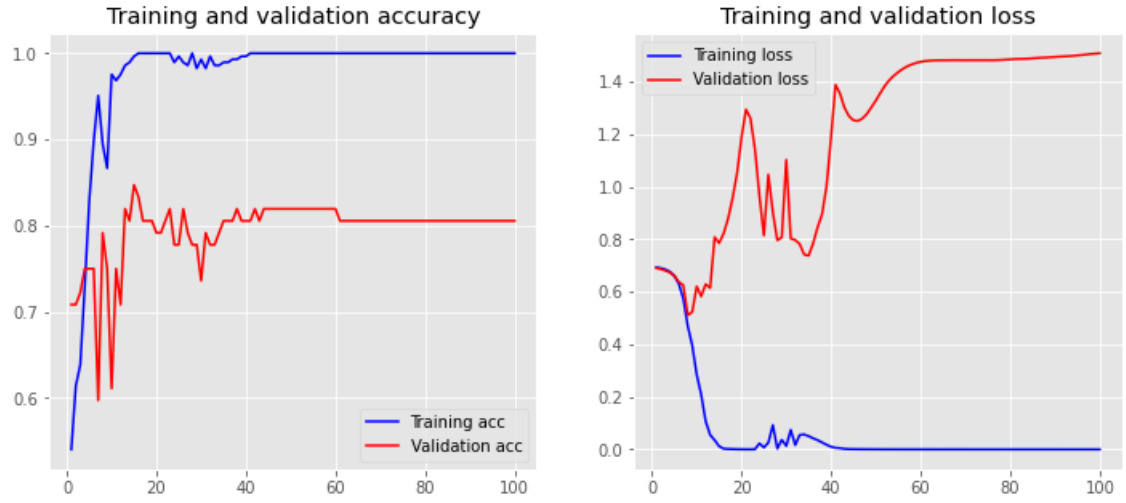



Figura 7.10 – Curvas de treinamento *Longformer* + DNN.

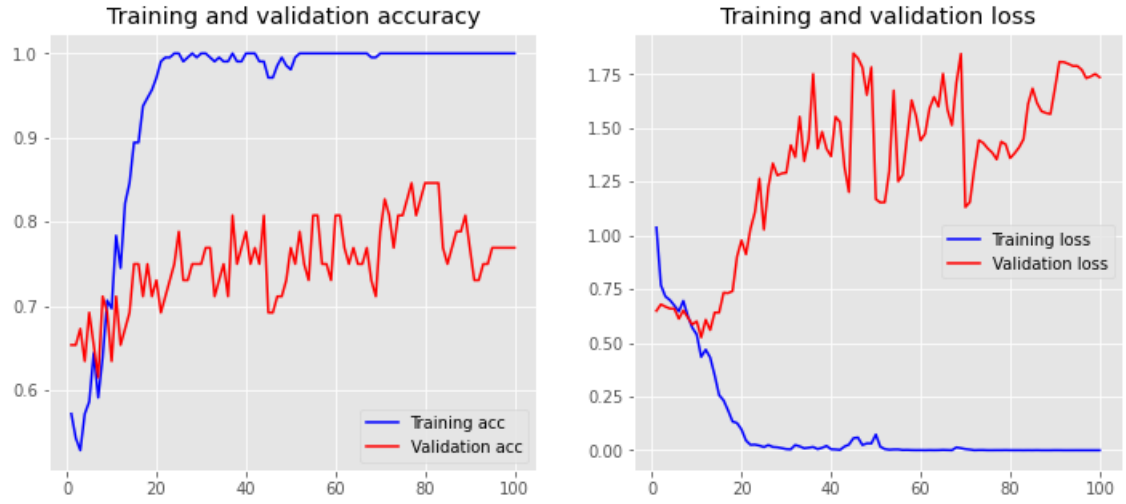



Figura 7.11 – Curvas de treinamento *Longformer* + CNN.

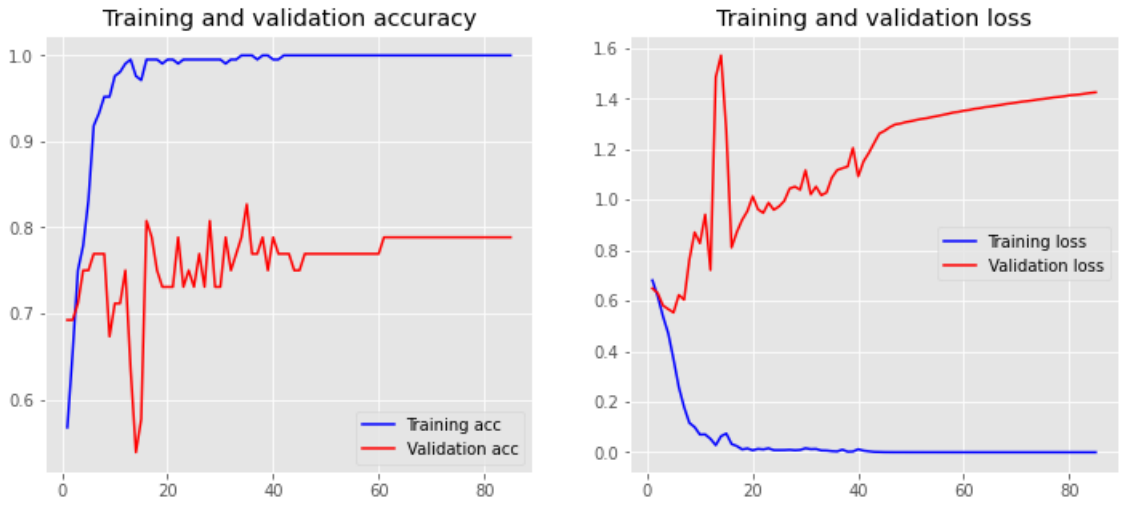



Figura 7.12 – Curvas de treinamento *Longformer* + LSTM.

Por fim, na curva de treinamento do LSTM (Figura 7.12), é possível observar uma queda na acurácia na 17ª época, seguida por uma melhora, atingindo seu pico na época 37.

Os resultados com métricas relacionadas podem ser visualizados na Tabela 7.3. Esta abordagem obteve boas acuracias, acima de 82% para todas as arquiteturas de implementação.

| **Modelo** | **Acurácia** | **F1 score** | **Precisão** | **Recall** |
|---|---|---|---|---|
| SNN | 0,8269 | 0,8754 | 0,7950 | 0,9773 |
| DNN | 0,8462 | 0,8776 | 0,8474 | 0,9123 |
| CNN | 0,8462 | 0,8776 | 0,8474 | 0,9123 |
| LSTM | 0,8269 | 0,8801 | 0,8571 | 0,9091 |

Tabela 7.3 – Resultados do *Longformer* combinado com os modelos de aprendizagem profunda para a classificação do *dataset* Preliminar.

### 7.1.4 Análise da Seção

Nesta subseção será discutido sobre o problema, os modelos utilizados e os resultados obtidos no contexto da classificação em PLN. A Tabela 7.4 mostra a visão geral dos resultados finais obtidos pelos experimentos da análise preliminar:

Ao analisar os dados, é possível observar um aspecto que dificulta a visualização das melhorias de precisão: os valores mudam em saltos de 1,9%. A quantidade de dados de teste rotulados é mínima e, como foi realizada uma segmentação de treinamento 80-20, a validação é feita com apenas 52 itens, o que explica os saltos.

Primeiramente, é válido destacar que já na análise preliminar foi possível superar os resultados de *baseline* obtidos por (SILVA, B. et al., 2021). Isso é bastante promissor, pois indica que a aplicação de aprendizagem profunda possui grande potencial para a classificação desse tipo de texto.

Tabela 7.4 – Comparativo dos resultados da análise preliminar. Fonte: (ROCHA. et al., 2022)

| **Model** | **Accuracy** | **F1 Score** | **Precision** | **Recall** |
|---|---|---|---|---|
| Baseline | | | | |
| Bag of words | 0,86 | 0,85 | 0,85 | 0,85 |
| Key words | 0,7808 | 0,7802 | 0,9358 | 0,6711 |
| Keras Word-embedding | | | | |
| NN | 0,8269 | 0,8620 | 0,8212 | 0,9129 |
| DNN | 0,8269 | 0,8650 | 0,8952 | 0,8447 |
| CNN | 0,8462 | 0,8756 | 1,0000 | 0,7803 |
| LSTM | 0,8269 | 0,8675 | 0,8276 | 0,9129 |
| Pre-trained Word2Vec Embeddings | | | | |
| NN | 0,8269 | 0,8545 | 0,8392 | 0,8712 |
| DNN | 0,7115 | 0,7794 | 0,7255 | 0,8485 |
| CNN | 0,8654 | 0,9083 | 08486 | 0,9773 |
| LSTM | 0,8846 | 0,9139 | 0,9056 | 0,9318 |
| Pre-trained Longformer | | | | |
| NN | 0,8269 | 0,8754 | 0,7950 | 0,9773 |
| DNN | 0,8462 | 0,8776 | 0,8474 | 0,9123 |
| CNN | 0,8462 | 0,8776 | 0,8474 | 0,9123 |
| LSTM | 0,8269 | 0,8801 | 0,8571 | 0,9091 |

Como já esperado, as curvas de aprendizado dos modelos que utilizaram SNN (Figuras 7.1, 7.5 e 7.9) apresentam um claro padrão de *underfitting*, ou seja, quando a rede não é capaz de modelar o problema corretamente. Mesmo assim, foi possível obter resultados interessantes, o que denota que, para parte do *dataset*, o critério de elegibilidade é relativamente simples de ser encontrado, e mesmo a rede mais simples é capaz de alcançar boa acurácia.

Todos os demais modelos apresentaram padrão de *overfitting*, quando o modelo se adapta aos dados de treinamento e não é capaz de generalizar novos dados a partir de certa época de treino. Isso pode ser notado quando a curva de perda de validação começa a crescer enquanto a curva de perda de treinamento (*training loss*) ainda decai. Por esse motivo não foram utilizados os pesos da última época de treino, e sim da época onde a acurácia de validação obteve seu valor máximo.

Quanto à transferência de aprendizado, é possível verificar que a maioria dos modelos de classificação testados apresentaram melhora na acurácia após o pré-treinamento. Em particular, a rede composta por *Word2Vec Embeddings* pré-treinados + rede LSTM obteve o melhor resultado para o conjunto de dados, com 88% de acurácia e mais de 90% de precisão.

Os resultados do *Longformer* superaram o modelo de *embeddings* simples e, embora satisfatórios, possivelmente podem ser melhorados com o ajuste fino do modelo, o que não foi possível com os recursos computacionais disponíveis. Mesmo assim, espera-se que com mais dados seja possível alcançar resultados superiores ao da *baseline*.

## 7.2 Resultados Pré-processamento de Dados

Conforme mencionado na Seção 6.4, foi realizada uma análise exploratória com o objetivo de identificar a melhor combinação de técnicas para a realização do pré-processamento dos dados. Para esta etapa foi utilizado o *dataset* Final, já objetivando a consolidação dos dados a serem utilizados na etapa final de classificação. Os dados foram processados seguindo os experimentos listados na Tabela 6.4. Os resultados desta etapa, considerando as métricas definidas na Seção 4.8, os tempos de treinamento, quantidade de *tokens* únicos (N_tokens) e o tamanho máximo de sentença após pré-processamento são apresentados na Tabela 7.5.

Tabela 7.5 – Resultados do pré-processamento.

| | **Acurácia** | **F1-score** | **Recall** | **Precisão** | **Tempo Méd.** | **N_tokens** | **Tam. máx.** |
|---|---|---|---|---|---|---|---|
| Base | 0,8978 | 0,8420 | 0,7909 | 0,9095 | 417,77 s | 23788 | 5636 |
| Xp1 | 0,8871 | 0,8159 | 0,7154 | 0,9733 | 414,72 s | 23768 | 5636 |
| Xp2 | 0,9032 | 0,8564 | 0,7719 | 0,9744 | 368,38 s | 20322 | 5629 |
| Xp3 | 0,9194 | 0,8773 | 0,7966 | 0,9872 | 386,65 s | 22121 | 4950 |
| Xp4 | 0,9086 | 0,8661 | 0,8085 | 0,9425 | 326,83 s | 18616 | 4950 |
| Xp5 | 0,9194 | 0,8768 | 0,7847 | 1,0000 | 257,96 s | 14319 | 4950 |
| Xp6 | 0,8978 | 0,8506 | 0,7966 | 0,9187 | 282,65 s | 16194 | 4950 |
| Xp7 | 0,9247 | 0,8846 | 0,7966 | 1,0000 | 210,32 s | 14212 | 2817 |
| Xp8 | 0,9247 | 0,8846 | 0,7966 | 1,0000 | 225,58 s | 16081 | 2726 |

Da Tabela 7.5, nota-se que os dois últimos experimentos (Xp7 e Xp8) apresentaram os melhores resultados para acurácia, *f1-score* e precisão, além de menores tempos de treinamento e menores tamanhos de sentença. Desta forma, ambas as técnicas apresentam ótimo desempenho e podem ser escolhidas como etapa de pré-processamento.

### 7.2.1 Análise da Seção

Fazendo uma análise detalhada dos resultados obtidos na etapa de pré-processamento, o Xp7 apresentou menor tempo de treinamento e menor número de *tokens* únicos (N_tokens), enquanto que o Xp8 apresentou menores tamanhos máximos. Embora o Xp7 tenha menor tempo de treinamento, a diferença com o Xp8 nesse mesmo quesito é tão pequena que não chega a ser significativa para tomada de decisão.

Levando agora em consideração os métodos e modelos a serem aplicados no decorrer do trabalho, os seguintes pontos se fazem relevantes para a análise de qual experimento se adequa melhor:

- *Keras Embedding*

  - Para o *Keras Embedding*, um maior número de *tokens* únicos representa apenas um *one-hot-encoding* de maior vocabulário, mas isso não aumenta necessariamente o tamanho da rede;
  - Já o tamanho da maior sentença modifica o tamanho do *input* da rede necessária para treinar, e isso também é refletido na quantidade de pesos da camada seguinte.

- *Word2Vec*
  - No *Word2Vec*, o maior número de *tokens* únicos significa um maior tempo para o pré-treino da rede. Como esse pré-treinamento só será feito uma vez, a interferência no sistema não é tão significativa. Além disso, o arquivo de pesos pré-treinados do *Word2Vec* será maior, e ele precisará ser carregado em memória para traduzir o *input* para a representação vetorial. Porém, este também pode ser descarregado da memória logo após a tradução, e não deve influenciar na memória gasta em treinamento;
  - Já o tamanho da maior sentença quase não interfere no treinamento do *Word2Vec*, porém aumenta o *input* da rede final, e a quantidade de dados carregados em memória após sua representação vetorial.

- *Longformer*
  - O *Longformer* utilizado por (BELTAGY et al., 2020) já foi pré-treinado anteriormente com uma quantidade muito superior de *tokens*, então, o número de *tokens* únicos é indiferente na análise;
  - O tamanho da maior sentença também interfere muito pouco, visto que os dois valores alcançados no Xp7 e Xp8 (2817 e 2726, respectivamente) estão abaixo dos 4096 do tamanho máximo permitido na rede. Porém, deve-se levar em consideração que, quando o modelo for aplicado para o uso do MCTI, possuir um modelo capaz de reduzir o tamanho do *input* de maneira criteriosa e eficaz resulta em menos perda informação sendo truncada ou desconsiderada, quando esta for maior que o limite máximo da rede.

Diante das questões levantadas, definiu-se que o modelo de pré-processamento mais adequado para esta etapa do projeto é a do experimento 8 (Xp8: XP4 + lematização + remoção de stopwords), pois possui menor tamanho de *input* para o treinamento e tende a reduzir mais o tamanho da entrada para o uso futuro, impedindo, assim, que informações sejam perdidas quando estas ultrapassarem o limite máximo de tamanho da rede.

## 7.3 Resultados da Análise Final

Agora serão apresentados os resultados da classificação do *dataset* Final, considerando também a etapa de pré-processamento através do modelo Xp8 adotado, com o qual foi possível realizar o treinamento completo das redes. Pelo fato de o novo *dataset* possuir 928 dados catalogados, era esperado resultados superiores aos encontrados na Seção 7.1, já que a rede pode aprender de maneira supervisionada através de mais exemplos.

### 7.3.1 Keras Embedding + Deep Learning

Os resultados de treinamento obtidos para a classificação do *dataset* Final utilizando o método de representação *Keras Embedding* associado aos modelos de aprendizagem profunda podem ser verificados na Tabela 7.6.

| Modelo | Acurácia | F1 score | Precisão | Recall |
|---|---|---|---|---|
| SNN | 0,9247 | 0,8846 | 1,0000 | 0,7966 |
| DNN | 0,8978 | 0,8441 | 0,9257 | 0,7781 |
| CNN | 0,9301 | 0,8991 | 0,9569 | 0,8518 |
| LSTM | 0,9301 | 0,8894 | 0,9554 | 0,8332 |

Tabela 7.6 – Resultados do *Keras Embedding* combinado com os modelos de aprendizagem profunda para a classificação do *dataset* Final.

A partir dos resultados apresentados, é possível observar que com o novo dataset houve um grande salto na acurácia de classificação. O resultado da rede *Keras* + SNN apresentou resultado superior à *Keras* + DNN, o que pode ser atribuído à quantidade superior de rodadas de treinamento que a rede teve, levando em consideração que ela também foi bastante treinada durante o pré-processamento. Destaca-se também o resultado das redes *Keras* + CNN e *Keras* + LSTM que alcançaram valores de 93% de acurácia, uma nova conquista para classificação desse tipo de texto.

### 7.3.2 Word2Vec Pré-treinado + Deep Learning

A Tabela 7.7 mostra os resultados obtidos com a aplicação do *Word2Vec* pré-treinado com os dados do *dataset* Final em conjunto com os modelos de aprendizagem profunda apresentados na Seção 6.2.

| Modelo | Acurácia | F1 score | Precisão | Recall |
|---|---|---|---|---|
| SNN | 0,8925 | 0,8382 | 0,9710 | 0,7415 |
| DNN | 0,9032 | 0,8652 | 0,8870 | 0,8518 |
| CNN | 0,9247 | 0,8842 | 0,9872 | 0,8085 |
| LSTM | 0,8978 | 0,8436 | 0,9581 | 0,7536 |

Tabela 7.7 – Resultados do *Word2Vec*, treinado com o *dataset* Final, combinado com os modelos de aprendizagem profunda para a classificação do *dataset* Final.

É possível observar que todos os modelos apresentaram resultados superiores à *baseline* (Subseção 3.1.1), porém inferiores aos obtidos com o *Keras* (Subseção 7.3.1).

Também foi realizado o experimento utilizando o modelo *Word2Vec* pré-treinado com a combinação dos *datasets* Preliminar, PPF e Final, em conjunto com os modelos de aprendizagem profunda. A Tabela 7.8 apresenta os resultados deste modelo.

| **Modelo** | **Acurácia** | **F1 score** | **Precisão** | **Recall** |
|---|---|---|---|---|
| SNN | 0,9301 | 0,8964 | 1,0000 | 0,8125 |
| DNN | 0,9247 | 0,8830 | 0,9688 | 0,8125 |
| CNN | 0,9355 | 0,9040 | 0,9878 | 0,8333 |
| LSTM | 0,9247 | 0,8844 | 0,9563 | 0,8229 |

Tabela 7.8 – Resultados do *Word2Vec*, treinado com o *dataset* Combinado, associado com os modelos de aprendizagem profunda para a classificação do *dataset* Final.

É possível verificar que houve um ganho significativo na acurácia dos modelos com a adição de mais dados para o treino do *Word2Vec*. Nota-se também que o modelo *Word2Vec* + CNN alcançou uma acurácia de 93.5% e *f1-score* superior a 90%, sendo este um alcance expressivo.

### 7.3.3 Longformer + Deep Learning

Na Tabela 7.9 é possível observar os resultados obtidos com o *Longformer* associado às redes de aprendizado profundo.

| **Modelo** | **Acurácia** | **F1 score** | **Precisão** | **Recall** |
|---|---|---|---|---|
| SNN | 0,8924 | 0,8554 | 0,8895 | 0,8377 |
| DNN | 0,9193 | 0,8762 | 0,8037 | 0,9762 |
| CNN | 0,9409 | 0,9069 | 0,8341 | 1,0000 |

Tabela 7.9 – Resultados do *Longformer* combinado com os modelos de aprendizagem profunda para a classificação do *dataset* Final.

O grande destaque é o resultado do modelo em conjunto com a rede CNN, que obteve o melhor valor de acurácia do trabalho com 94%, *f1-score* superior a 90% e precisão de 100%.

### 7.3.4 Análise da Seção

Os resultados da classificação do *dataset* Final, considerando também a etapa de pré-processamento, são expostos na Tabela 7.10, que apresenta, além das métricas de performance, os tempos gastos para o treino de cada época e o tempo para a validação dos resultados, ou seja, para que o modelo classifique os 20% relativos à validação do *dataset* Final.

É possível constatar que estes resultados superaram, de fato, os resultados alcançados na análise preliminar (Seção 7.1), com melhores acurácias obtidas de 94% para o modelo

*Longformer* + CNN, 93.5% com o modelo *Word2Vec* + CNN e 93% com os modelos *Keras* + CNN e *Keras* + LSTM. Essa melhoria pode ser atribuída principalmente à maior quantidade de dados catalogados utilizada para o treinamento supervisionado. Outro fator importante é a qualidade superior dos dados, que foram obtidos através do uso de *scripts* de raspagem (Seção 2.1) mais atualizados.

Uma observação interessante é de que os *Word Embeddings* treinados de maneira não supervisionada através do *Word2Vec* já não foram necessariamente superiores aos do *Keras*, como pode ser visto na Tabela 7.7, indicando que sem a utilização de dados adicionais, os *embeddings* treinados com *Word2Vec* não são superiores aos *embeddings* simples treinados junto com a rede. Isso se dá pelo fato do novo *dataset* possuir mais dados catalogados, melhorando assim, a aprendizagem conjunta de maneira supervisionada.

Entretanto, com a combinação dos *datasets* para a realização do pré-treino dos pesos *Word2Vec*, os modelos apresentaram ótimos resultados, como pode ser visto na Tabela 7.8.

Outro destaque importante é a diferença no tempo de treinamento dos modelos. Os modelos que utilizaram *Keras embedding* levaram em média menos de 1s por época, que associado com os tempos de desalocação de memória e cálculo das métricas, resultou em uma duração média de 25 minutos para o treinamento das redes. Os modelos que utilizam *Word2Vec* apresentaram tempos de treino levemente superiores, ainda abaixo de 2s por época, que resultaram em um tempo de treinamento médio de 35 minutos. Já os modelos utilizando o *Longformer* que, conforme comentado na Seção 6.5, precisaram ser treinados utilizando CPU e apresentaram tempo médio por época de 55s, o que levou a um tempo de treinamento médio de 5 horas.

Mesmo com a utilização do ambiente da alta RAM para viabilizar o treinamento dos modelos associados ao *Longformer*, ainda foram necessárias otimizações como a redução do tamanho do *batch* e o treinamento de poucas épocas de cada vez. Com isso foi possível realizar o treinamento dos modelos, porém infelizmente o modelo associado à rede LSTM não foi capaz de aprender. Conforme comentado na Subseção 7.3.4, o tempo médio de treino foi de 5 horas, o que tornou difícil o processo de ajustes e validação, principalmente levando em consideração o limite mensal de uso do recurso.

Apesar do tempo de treinamento ser bem superior para o modelo do *Longformer*, o tempo de execução do sistema (tempo de validação) não se afasta muito dos outros modelos, o que não necessariamente tornaria uma aplicação com esse modelo mais lenta em comparação com os outros.

Tabela 7.10 – Resultados da análise final.

| Modelo | Acurácia | F1-score | Recall | Precisão | Tempo época | Tempo validação |
|---|---|---|---|---|---|---|
| Keras + SNN | 0,9247 | 0,8846 | 0,7966 | 1,000 | 0,2s | 0,7s |
| Keras + DNN | 0,8978 | 0,8441 | 0,7781 | 0,9257 | 1s | 1,4s |
| Keras + CNN | 0,9301 | 0,8991 | 0,8518 | 0,9569 | 0,4s | 1,1s |
| Keras + LSTM | 0,9301 | 0,8894 | 0,8332 | 0,9554 | 1,4s | 2s |
| Word2Vec + SNN | 0,9301 | 0,8964 | 0,8125 | 1,0000 | 1,4s | 1,2s |
| Word2Vec + DNN | 0,9247 | 0,8830 | 0,8125 | 0,9688 | 2s | 6,8s |
| **Word2Vec + CNN** | **0,9355** | 0,9040 | 0,8333 | 0,9878 | 1,9s | 3,4s |
| Word2Vec + LSTM | 0,9247 | 0,8844 | 0,8229 | 0,9563 | 2,6s | 14,3s |
| Longformer + SNN | 0,8924 | 0,8554 | **0,8895** | 0,8377 | 28s | 2,5s |
| Longformer + DNN | 0,9193 | 0,8762 | 0,8037 | 0,9762 | 81s | 8,4s |
| **Longformer + CNN** | **0,9409** | **0,9069** | 0,8341 | **1,0000** | 57s | 4,5s |

# 8 Discussões e Considerações Finais

A aplicação de modelos de processamento de linguagem natural e aprendizagem profunda para a classificação de textos de oportunidades de financiamento para o PPF foi o objetivo principal desta pesquisa. Esses textos longos, não estruturados, não uniformes e escassos representaram um desafio claro onde se fez necessário a utilização de estratégias estado-da-arte para realização de transferência de aprendizado e treinamento das redes.

A análise preliminar realizada permitiu validar que a utilização de representações vetoriais contextualizadas, associadas às redes de aprendizagem profunda, foi capaz de realizar a classificação desse tipo de texto e ainda ultrapassar os resultados da baseline obtida por (SILVA, B. et al., 2021). A partir dela foi possível identificar melhorias para o sistema e também ajustar os modelos de maneira a maximizar os resultados obtidos.

Uma estratégia que resultou em excelentes melhorias para o sistema foi a adição de um processo estruturado de pré-processamento. Os ensaios elaborados através da combinação de várias técnicas conhecidas de processamento de linguagem natural e normalização de textos foram capazes de incrementar a acurácia da classificação em pelo menos 2% (Tabela 7.5).

A estratégia aplicada no experimento nomeado Xp8, selecionada pelos critérios descritos na Subseção 7.2.1, consistiu na combinação das técnicas de expansão de contrações, padronização dos caracteres para minúsculas, remoção da pontuação, lematização e remoção de stopwords. Em adição a maximizar as métricas de avaliação da performance da rede, com ela foi possível reduzir o tamanho máximo de sentença para menos da metade, otimizando o tamanho da rede necessária para a aplicação, além de reduzir significativamente a quantidade de tokens únicos que precisam ser aprendidos pela rede.

Além da melhoria no pré-processamento, também foi observada a necessidade de uma quantidade maior de dados para o treinamento da rede, e para isso foi introduzido o *dataset* Final. Em cima desse novo *dataset* pré-processado, foram realizadas novas rodadas de treinamento onde foi possível alcançar um novo patamar de resultados, excedendo a barreira dos 90% de acurácia em grande parte dos modelos treinados. É importante destacar o modelo *Longformer* + CNN que atingiu 94% de acurácia com 100% de precisão. Outros modelos também obtiveram bom desempenho fazendo uso de menos recursos computacionais como o *Word2Vec* + CNN com 93,55%, o *Keras Embedding* + CNN e o *Keras Embedding* + LSTM com 93% de acurácia. Outro detalhe importante a ser salientado é o desempenho da arquitetura CNN modelada, que esteve correlacionado aos melhores resultados obtidos na pesquisa.

O pré-treino dos pesos *Word2Vec* se mostrou uma etapa fundamental para o bom desempenho dos modelos. Foi possível observar um ganho significativo no entendimento

dos textos quando comparado com os modelos utilizando *Keras Embedding*. Porém, como comentado na Subseção 7.3.4, esse ganho só foi observado quando o pré-treino foi realizado com uma quantidade superior de dados, seja na análise preliminar ou na etapa de análise final. Esse fato corrobora com a tendência dos modelos recentes serem pré-treinados com cada vez mais dados com o objetivo de maximizar os resultados.

Para a definição do melhor modelo para o uso do MCTI é necessário analisar os melhores modelos e compreender suas limitações. O modelo que obteve os melhores resultados foi o *Longformer* associado à rede CNN com 94% de acurácia. Porém, conforme apresentado na Seção 6.5, o *Longformer* necessita de muita memória RAM para ser executado e o peso do modelo (Tabela 6.5) é superior ao máximo permitido pelos repositórios do *Github* ou *Hugging Face*.

O segundo melhor modelo encontrado é o *Word2Vec* + CNN. Ele não apresenta nenhum limitante de performance e possui um tamanho de apenas 85,2 Mb, conforme apresentado na Tabela 6.5. Por esse motivo, este modelo foi utilizado para o protótipo da aplicação de classificação, descrito na Seção 6.6.

Outro aspecto muito relevante a ser analisado são as oportunidades que foram classificadas incorretamente pelo modelo. A Tabela 8.1 apresenta as colunas *opo_texto* e *opo_texto_ele* de cinco oportunidades que foram erroneamente classificadas pelo modelo. Com esse detalhamento é possível observar que alguns dos textos não apresentam informações suficientes para a tomada de decisão correta, o que significa que existe uma limitação relacionada à qualidade dos dados fornecidos para a rede.

Tabela 8.1 – Oportunidades com textos inconclusivos

| **Elegibilidade** | **opo_texto** | **opo_texto_ele** |
|---|---|---|
| Elegível | Joint Call for Research Proposals on Ã¢â,¬Å“Supporting the Protection of Biodiversity and Ecosystems across Land and SeaÃ¢â,¬ | Joint Call for Research Proposals on Ã¢â,¬Å“Supporting the Protection of Biodiversity and Ecosystems across Land and SeaÃ¢â,¬ |
| | | Continua... |

**Tabela 8.1**

| Elegibilidade | opo_texto | opo_texto_ele |
|---|---|---|
| Elegível | This one-off call aims to support multidisciplinary teams to help improve our understanding of the biological significance of SARS-CoV-2 variants, focused on laboratory investigations in immunology, virology or structural biology. | nan |
| Elegível | We are seeking proposals along the pharmaceutical R&D value chain that can tangibly improve sustainability aspects in R&D. The proposals should address our search scope and suggest a way forward. | We are seeking proposals along the pharmaceutical R&D value chain that can tangibly improve sustainability aspects in R&D. The proposals should address our search scope and suggest a way forward. |
| Elegível | Science and technology are at the heart of everything we do, driving innovations that enable us to contribute to a sustainable future. In this endeavor, we are seeking research proposals in the field of Green Chemistry. | Science and technology are at the heart of everything we do, driving innovations that enable us to contribute to a sustainable future. In this endeavor, we are seeking research proposals in the field of Green Chemistry. |

Continua...

**Tabela 8.1**

| Elegibilidade | opo_texto | opo_texto_ele |
|---|---|---|
| Elegível | This call provides funding for teams of researchers working across any discipline of relevance to mental health science. They will investigate the causal mechanisms underpinning effective interventions for anxiety, depression, and/or psychosis, to inform the development of new and improved early interventions. | nan |
| *Fonte: Dataset Final.* | | |

O cenário apresentado na Tabela 8.1 corresponde a menos de 5% dos dados do *dataset Final*, portanto é possível concluir que os modelos ainda podem se beneficiar de mais dados de treinamento e mais otimizações, porém estarão limitados à qualidade da entrada.

Outro detalhe importante de ser analisado é o valor obtido para a precisão em alguns dos modelos. Apesar do resultado positivo não é possível garantir que o sistema possui 100% de precisão, principalmente pelo motivo destacado anteriormente relativo a qualidade dos dados mas também pela pouca quantidade de dados utilizados para a validação.

# 9 Conclusão e Trabalhos Futuros

A classificação de textos é um problema tradicional em processamento de linguagem natural, sendo este um tema que vem ganhando destaque e grandes avanços nos últimos anos devido à crescente popularização da inteligência artificial. Esses avanços despertaram o interesse do MCTI em automatizar os processos relacionados ao seu portfólio de produtos financeiros (Capítulo 2).

Para desenvolver um sistema capaz de identificar e recomendar oportunidades de financiamento no contexto do PPF de maneira automática, é de extrema importância a implementação de um modelo capaz de filtrar essas oportunidades quanto a sua elegibilidade para projetos brasileiros. Dessa forma, neste trabalho é proposto uma solução para o problema de classificação dos textos de oportunidades, considerando suas características: textos longos, não estruturados, não uniformes e escassos.

A partir de uma análise sistemática do estado de arte (Seção 3.2) foi possível identificar técnicas de aprendizagem profunda, representação contextual de textos e transferência de aprendizado que motivaram o desenvolvimento dos modelos utilizados. O projeto utilizou, como mecanismo de representação textual, *Word Embeddings* com o *Keras* e *Word2Vec* e *document-long embeddings* com o *Longformer*, e como rede neural para a classificação utilizou-se modelos SNN, DNN, CNN e LSTM. Foram realizados experimentos através de uma análise cruzada das técnicas e modelos para validar o desempenho e avaliar qual o mais indicado para a aplicação.

Os dados utilizados para o treinamento desses modelos (Capítulo 5) foram adquiridos através scripts de raspagem de dados, aplicados em plataformas online de diversas instituições de financiamento. Esses dados apresentavam bastante ruído e problemas de codificação de texto, e por isso, necessitaram da aplicação de um pré-processamento. Para encontrar as melhores técnicas a serem aplicadas, foram elaborados experimentos (Tabela 6.4) que permitiram reduzir o tamanho máximo de sentença para menos da metade, reduzir significativamente a quantidade de tokens únicos e aumentar o desempenho da classificação em pelo menos 2%.

Durante o desenvolvimento da pesquisa foram utilizados os modelos de transferência de aprendizado *Word2Vec* e *Longformer*. Devido a limitação de poder computacional, não foi possível realizar um pré-treino ou ajuste fino dos pesos do *Longformer*, mesmo assim, o modelo se mostrou altamente eficiente na realização desse tipo de classificação, alcançando 94% de acurácia na arquitetura utilizando a CNN.

O *Word2Vec* também atingiu um ótimo desempenho, alcançando 93,55% de acurácia na arquitetura com CNN. Com ele foi possível realizar um pré-treino utilizando textos de

oportunidades não catalogadas através do seu aprendizado não supervisionado. Quando comparado com os pesos treinados apenas com os dados catalogados do *dataset* Final (Subseção 7.3.2) foi possível observar um ganho de até 3,5% em acurácia devido ao pré-treinamento.

Apesar da arquitetura *Longformer* + CNN ter apresentado o melhor resultado de classificação, foi observado a necessidade de um grande poder computacional para o uso do modelo e por esse motivo a arquitetura *Word2Vec* + CNN foi selecionada para compor o protótipo de aplicação (Seção 6.6) disponibilizada para uso do MCTI. Isso ressalta a importância da abordagem incremental realizada como metodologia do projeto, que permitiu a descoberta de um modelo adequado aos requisitos de implementação.

Nesta pesquisa, a classificação das oportunidades de financiamento de projetos associada ao PPF atingiu um novo patamar de resultados, superando (SILVA, B. et al., 2021). Estes resultados inspiraram a publicação do artigo (ROCHA. et al., 2022) e contribuíram para o estreitamento das relações entre governo e universidade.

Para trabalhos futuros sugere-se:

- Realização de melhorias na aquisição e tratamento dos dados para garantir a qualidade das informações e melhorar o desempenho do sistema.
- Comparativo da estratégia de pré-treinamento *Word2Vec* com CNN para a classificação de outros *datasets* da literatura.
- Com a disponibilidade de maior poder computacional, realizar um ajuste fino do modelo *Longformer* e o desenvolvimento de mais otimizações para viabilizar seu uso pelo MCTI.

# Referências


AI2. **Semantic Scholar - About Us**. Allen Institute for AI, 2023. Disponível em: <https://www.semanticscholar.org/about>. Acesso em: 16 mar. 2023. Citado na p. 30.

AINSLIE, J.; ONTANON, S.; ALBERTI, C.; CVICEK, V.; FISHER, Z.; PHAM, P.; RAVULA, A.; SANGHAI, S.; WANG, Q.; YANG, L. **ETC: Encoding Long and Structured Inputs in Transformers**. arXiv, 2020. DOI: 10.48550/ARXIV.2004.08483. Disponível em: <https://arxiv.org/abs/2004.08483>. Acesso em: 16 mar. 2023. Citado nas pp. 34, 35.

ALACHRAM, H.; CHEREDA, H.; BEISSBARTH, T.; WINGENDER, E.; STEGMAIER, P. **Text mining-based word representations for biomedical data analysis and machine learning tasks**. Dez. 2020. DOI: 10.1101/2020.12.09.417733. Citado na p. 34.

ALLAHYARI, M.; POURIYEH, S.; ASSEFI, M.; SAFAEI, S.; TRIPPE, E. D.; GUTIERREZ, J. B.; KOCHUT, K. **A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques**. arXiv, 2017. DOI: 10.48550/ARXIV.1707.02919. Disponível em: <https://arxiv.org/abs/1707.02919>. Acesso em: 16 mar. 2023. Citado na p. 38.

BELTAGY, I.; PETERS, M. E.; COHAN, A. **Longformer: The Long-Document Transformer**. arXiv, 2020. DOI: 10.48550/arXiv.2004.05150. Disponível em: <https://arxiv.org/abs/2004.05150>. Acesso em: 16 mar. 2023. Citado nas pp. 35, 52, 53, 68, 92.

BENTO, C. **Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis**. 2021. Disponível em: <https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141>. Acesso em: 20 jan. 2023. Citado nas pp. 46, 47.

BOOK, D. L. **Capítulo 51 – Arquitetura de Redes Neurais Long Short Term Memory (LSTM)**. 2022a. Disponível em: <https://www.deeplearningbook.com.br/arquitetura-de-redes-neurais-long-short-term-memory/>. Acesso em: 14 dez. 2022. Citado nas pp. 49, 50.

BOOK, D. L. **Capítulo 6 – O Perceptron – Parte 1**. 2022b. Disponível em: <https://www.deeplearningbook.com.br/o-perceptron-parte-1/#:~:text=Perceptron%5C%20%5C%C3%5C%A9%5C%20uma%5C%20rede%5C%20neural,os%5C%20dados%5C%20de%5C%20entrada%5C%20fornecidos.>. Acesso em: 20 jan. 2023. Citado na p. 46.

BRASIL. **Ministério de ciência, tecnologia e inovações. portfólio de produtos financeiros**. Ministério de Ciência, Tecnologia e Inovações, 2019. Disponível em: <https://ppf.mcti.gov.br/>. Acesso em: 16 mar. 2023. Citado na p. 29.

BROWN, T.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J. D.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A. et al. Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020. Citado nas pp. 33, 36.

BULK, L. M. van den; BOUZEMBRAK, Y.; GAVAI, A.; LIU, N.; HEUVEL, L. J. van den; MARVIN, H. J. Automatic classification of literature in systematic reviews on food safety using machine learning. **Current Research in Food Science**, Elsevier, v. 5, p. 84–95, 2022. Citado na p. 36.

CHOWDHURY, S. A.; STEPANOV, E. A.; RICCARDI, G. Predicting User Satisfaction from Turn-Taking in Spoken Conversations. In: PROC. Interspeech 2016. 2016. P. 2910–2914. DOI: 10.21437/Interspeech.2016-859. Citado na p. 34.

CHURCH, K. W.; CHEN, Z.; MA, Y. Emerging trends: A gentle introduction to fine-tuning. **Natural Language Engineering**, Cambridge University Press, v. 27, n. 6, p. 763–778, 2021. DOI: 10.1017/S1351324921000322. Citado na p. 54.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **CoRR**, abs/1810.04805, 2018. arXiv: 1810.04805. Disponível em: <http://arxiv.org/abs/1810.04805>. Acesso em: 16 mar. 2023. Citado nas pp. 33, 36.

DINTER, R. van; CATAL, C.; TEKINERDOGAN, B. A decision support system for automating document retrieval and citation screening. **Expert Systems with Applications**, Elsevier, v. 182, p. 115261, 2021. Citado na p. 36.

DIOGO NOGARE. **Performance de Machine Learning – Matriz de Confusão**. 2020. Disponível em: <https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/>. Acesso em: 10 jan. 2023. Citado na p. 56.

DO, C. B.; NG, A. Y. Transfer learning for text classification. In: WEISS, Y.; SCHÖLKOPF, B.; PLATT, J. (Ed.). **Advances in Neural Information Processing Systems**. MIT Press, 2005. v. 18. Disponível em: <https://proceedings.neurips.cc/paper/2005/file/bf2fb7d1825a1df3ca308ad0bf48591e-Paper.pdf>. Acesso em: 16 mar. 2023. Citado na p. 36.

FACE, H. **Hugging Face Spaces**. HuggingFace.co, 2023. Disponível em: <https://huggingface.co/docs/hub/spaces>. Acesso em: 16 mar. 2023. Citado na p. 80.

FCHOLLET. **Introduction to Keras for Researchers**. Keras.io, 2020. Disponível em: <https://keras.io/getting_started/intro_to_keras_for_researchers/>. Acesso em: 16 mar. 2023. Citado na p. 54.

FEI-FEI, L.; FERGUS, R.; PERONA, P. A Bayesian approach to unsupervised one-shot learning of object categories. In: IEEE. PROCEEDINGS ninth IEEE international conference on computer vision. 2003. P. 1134–1141. Citado na p. 35.

GOLDBERG, Y.; LEVY, O. **word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method**. arXiv, 2014. DOI: 10.48550/ARXIV.1402.3722. Disponível em: <https://arxiv.org/abs/1402.3722>. Acesso em: 16 mar. 2023. Citado na p. 43.

GRADIO. **Gradio**. Gradio.app, 2023. Disponível em: <https://gradio.app/quickstart/>. Acesso em: 16 mar. 2023. Citado na p. 80.

GRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 1st: O'Reilly Media, Inc., 2017. ISBN 1491962291. Citado na p. 33.

GU, J.; ZHU, M.; ZHOU, Z.; ZHANG, F.; LIN, Z.; ZHANG, Q.; BRETERNITZ, M. Implementation and Evaluation of Deep Neural Networks (DNN) on Mainstream Heterogeneous Systems. In: PROCEEDINGS of 5th Asia-Pacific Workshop on Systems. Beijing, China: Association for Computing Machinery, 2014. (APSys '14). ISBN 9781450330244. DOI: 10.1145/2637166.2637229. Disponível em: <https://doi.org/10.1145/2637166.2637229>. Acesso em: 16 mar. 2023. Citado na p. 47.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3rd ed – Morgan Kaufmann Publishers. ISBN 978-0-12-381479-1. 2012. Citado na p. 38.

HAZEN, T. J. MCE Training Techniques for Topic Identification of Spoken Audio Documents. **IEEE Transactions on Audio, Speech, and Language Processing**, v. 19, n. 8, p. 2451–2460, 2011. DOI: 10.1109/TASL.2011.2139207. Citado na p. 34.

JING, R. A Self-attention Based LSTM Network for Text Classification. **Journal of Physics: Conference Series**, v. 1207, p. 012008, abr. 2019. DOI: 10.1088/1742-6596/1207/1/012008. Citado na p. 35.

JUNIOR, J. R. F. **Redes Neurais Recorrentes — LSTM**. 2019. Disponível em: <https://medium.com/@web2ajax/redes-neurais-recorrentes-lstm-b90b720dc3f6>. Acesso em: 2 jan. 2023. Citado na p. 49.

KERAS. **About Keras**. Keras.io, 2022. Disponível em: <https://keras.io/about/>. Acesso em: 16 mar. 2023. Citado na p. 54.

KESIRAJU, S.; BURGET, L.; SZŐKE, I.; ČERNOCKÝ, J. Learning document representations using subspace multinomial model. Versão inglesa. In: PROCEEDINGS of Interspeech 2016. San Francisco, US: International Speech Communication Association, 2016. P. 700–704. ISBN 978-1-5108-3313-5. DOI: 10.21437/Interspeech.2016-1634. Disponível em: <https://www.fit.vut.cz/research/publication/11269>. Acesso em: 16 mar. 2023. Citado na p. 34.

KIESEL, J.; MESTRE, M.; SHUKLA, R.; VINCENT, E.; ADINEH, P.; CORNEY, D.; STEIN, B.; POTTHAST, M. SemEval-2019 Task 4: Hyperpartisan News Detection. In: PROCEEDINGS of the 13th International Workshop on Semantic Evaluation. Minneapolis, Minnesota, USA: Association for Computational Linguistics, jun. 2019. P. 829–839. DOI: 10.18653/v1/S19-2145. Disponível em: <https://aclanthology.org/S19-2145>. Acesso em: 16 mar. 2023. Citado na p. 53.

KIM, Y. Convolutional Neural Networks for Sentence Classification. In: PROCEEDINGS of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, out. 2014. P. 1746–1751. DOI: 10.3115/v1/D14-1181. Disponível em: <https://aclanthology.org/D14-1181>. Acesso em: 16 mar. 2023. Citado nas pp. 33, 49.

KITCHENHAM, B. A.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. English. Jul. 2007. Disponível em: <https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf>. Acesso em: 16 mar. 2023. Citado na p. 30.

KONTONATSIOS, G.; SPENCER, S.; MATTHEW, P.; KORKONTZELOS, I. Using a neural network-based feature extraction method to facilitate citation screening for systematic reviews. **Expert Systems with Applications: X**, Elsevier, v. 6, p. 100030, 2020. Citado na p. 36.

KOWSARI; MEIMANDI, J.; HEIDARYSAFA; MENDU; BARNES; BROWN. Text Classification Algorithms: A Survey. **Information**, MDPI AG, v. 10, n. 4, p. 150, abr. 2019. DOI: 10.3390/info10040150. Citado nas pp. 47, 48.

LI, J.; WANG, C.; FANG, X.; YU, K.; ZHAO, J.; WU, X.; GONG, J. Multi-Label Text Classification via Hierarchical Transformer-CNN. In: 2022 14th International Conference on Machine Learning and Computing (ICMLC). Guangzhou, China: Association for Computing Machinery, 2022. (ICMLC 2022), p. 120–125. ISBN 9781450395700. DOI: 10.1145/3529836.3529912. Disponível em: <https://doi.org/10.1145/3529836.3529912>. Acesso em: 16 mar. 2023. Citado na p. 33.

LI, P.; ZHONG, P.; MAO, K.; WANG, D.; YANG, X.; LIU, Y.; YIN, J.; SEE, S. ACT: an Attentive Convolutional Transformer for Efficient Text Classification. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 35, n. 15, p. 13261–13269, mai. 2021. DOI: 10.1609/aaai.v35i15.17566. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/17566>. Acesso em: 16 mar. 2023. Citado na p. 34.

LIN, R.; CHENG, L.; DENG, J.; WANG, T. Multi-Level Feature Fusion Method for Long Text Classification. In: 2022 14th International Conference on Machine Learning and Computing (ICMLC). Guangzhou, China: Association for Computing Machinery, 2022. (ICMLC 2022), p. 532–538. ISBN 9781450395700. DOI: 10.1145/3529836.

3529938. Disponível em: <https://doi.org/10.1145/3529836.3529938>. Acesso em: 16 mar. 2023. Citado na p. 35.

LIN, Z.; FENG, M.; SANTOS, C. N. d.; YU, M.; XIANG, B.; ZHOU, B.; BENGIO, Y. **A Structured Self-attentive Sentence Embedding**. arXiv, 2017. DOI: 10.48550/ARXIV.1703.03130. Disponível em: <https://arxiv.org/abs/1703.03130>. Acesso em: 16 mar. 2023. Citado na p. 32.

LUQUE, J.; SEGURA, C.; SÁNCHEZ, A.; UMBERT, M.; GALINDO, L. A. The Role of Linguistic and Prosodic Cues on the Prediction of Self-Reported Satisfaction in Contact Centre Phone Calls. In: PROC. Interspeech 2017. 2017. P. 2346–2350. DOI: 10.21437/Interspeech.2017-424. Disponível em: <http://dx.doi.org/10.21437/Interspeech.2017-424>. Acesso em: 16 mar. 2023. Citado na p. 34.

MAAS, A. L.; DALY, R. E.; PHAM, P. T.; HUANG, D.; NG, A. Y.; POTTS, C. Learning Word Vectors for Sentiment Analysis. In: PROCEEDINGS of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, jun. 2011. P. 142–150. Disponível em: <https://aclanthology.org/P11-1015>. Acesso em: 16 mar. 2023. Citado na p. 53.

MAATEN, L. van der; HINTON, G. E. Visualizing Data using t-SNE. **Journal of Machine Learning Research**, v. 9, p. 2579–2605, 2008. Citado na p. 74.

MCCANN, B.; BRADBURY, J.; XIONG, C.; SOCHER, R. Learned in Translation: Contextualized Word Vectors. In: NIPS. 2017. Citado na p. 32.

MCCULLOCH, W.; PITTS, W. A Logical Calculus of Ideas Immanent in Nervous Activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 127–147, 1943. Citado na p. 46.

MEINZER, S.; JENSEN, U.; THAMM, A.; HORNEGGER, J.; ESKOFIER, B. Can machine learning techniques predict customer dissatisfaction? A feasibility study for the automotive industry. **Artif. Intell. Res.**, v. 6, p. 80–90, 2017. Citado na p. 34.

MELO, M.; FARIA, A. V.; WEIGANG, L.; NERY, A.; OLIVEIRA, F.; BARREIRO, I.; CELESTINO, V. Few-shot Approach for Systematic Literature Review Classifications. In: p. 33–44. DOI: 10.5220/0011526400003318. Citado na p. 30.

MIKOLOV, T.; KARAFIÁT, M.; BURGET, L.; CERNOCKÝ, J.; KHUDANPUR, S. Recurrent Neural Network Based Language Model. In: PROCEEDINGS of the 11th Annual Conference of the International Speech Communication Association. Makuhari, Chiba, Japan: ISCA, 2010. (INTERSPEECH 2010), p. 1045–1048. Disponível em: <https://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf>. Acesso em: 16 mar. 2023. Citado na p. 49.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; DEAN, J. Distributed Representations of Words and Phrases and Their Compositionality. In: PROCEEDINGS of the 26th International Conference on Neural Information Processing Systems - Volume 2. Lake Tahoe, USA: Curran Associates Inc., 2013. (NIPS'13), p. 3111–3119. Citado nas pp. 32, 43, 44.

MILLER, E. G.; MATSAKIS, N. E.; VIOLA, P. A. Learning from one example through shared densities on transforms. In: PROCEEDINGS IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662). 2000. v. 1, p. 464–471. Citado na p. 35.

MINAEE, S.; KALCHBRENNER, N.; CAMBRIA, E.; NIKZAD, N.; CHENAGHLU, M.; GAO, J. Deep Learning–Based Text Classification: A Comprehensive Review. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 3, abr. 2021. ISSN 0360-0300. DOI: 10.1145/3439726. Disponível em: <https://doi.org/10.1145/3439726>. Acesso em: 16 mar. 2023. Citado na p. 33.

MOHAMMED, A. H.; ALI, A. H. Survey of BERT (Bidirectional Encoder Representation Transformer) types. In: IOP PUBLISHING. JOURNAL of Physics: Conference Series. 2021. v. 1963, p. 012173. Citado na p. 54.

NIELSEN, M. A. **Neural networks and deep learning**. Determination press San Francisco, USA, 2015. v. 25, cap. 5. Citado na p. 47.

PAN, S. J.; TSANG, I. W.-H.; KWOK, J. T.-Y.; YANG, Q. Domain Adaptation via Transfer Component Analysis. **IEEE Transactions on Neural Networks**, v. 22, p. 199–210, 2011. Citado na p. 36.

PAN, S. J.; YANG, Q. A Survey on Transfer Learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191. Citado na p. 36.

PAPPAGARI, R.; VILLALBA, J.; DEHAK, N. Joint Verification-Identification in end-to-end Multi-Scale CNN Framework for Topic Identification. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018. P. 6199–6203. DOI: 10.1109/ICASSP.2018.8461673. Citado na p. 34.

PARK, Y.; GATES, S. C. Towards Real-Time Measurement of Customer Satisfaction Using Automatically Generated Call Transcripts. In: PROCEEDINGS of the 18th ACM Conference on Information and Knowledge Management. Hong Kong, China: Association for Computing Machinery, 2009. (CIKM '09), p. 1387–1396. ISBN 9781605585123. DOI: 10.1145/1645953.1646128. Disponível em: <https://doi.org/10.1145/1645953.1646128>. Acesso em: 16 mar. 2023. Citado na p. 34.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. **On the difficulty of training Recurrent Neural Networks**. arXiv, 2012. DOI: 10.48550/ARXIV.1211.5063. Disponível em: <https://arxiv.org/abs/1211.5063>. Acesso em: 16 mar. 2023. Citado na p. 49.

PENNINGTON, J.; SOCHER, R.; MANNING, C. GloVe: Global Vectors for Word Representation. In: PROCEEDINGS of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, out. 2014. P. 1532–1543. DOI: 10.3115/v1/D14-1162. Disponível em: <https://aclanthology.org/D14-1162>. Acesso em: 16 mar. 2023. Citado nas pp. 32, 43.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. **Deep contextualized word representations**. arXiv, 2018. DOI: 10.48550/ARXIV.1802.05365. Disponível em: <https://arxiv.org/abs/1802.05365>. Acesso em: 16 mar. 2023. Citado na p. 32.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. Improving language understanding by generative pre-training, 2018. Citado na p. 33.

RAINA, R.; BATTLE, A.; LEE, H.; PACKER, B.; NG, A. Y. Self-Taught Learning: Transfer Learning from Unlabeled Data. In: PROCEEDINGS of the 24th Int. Conf. on Machine Learning. Corvalis, Oregon, USA: Association for Computing Machinery, 2007. (ICML '07), p. 759–766. ISBN 9781595937933. DOI: 10.1145/1273496.1273592. Disponível em: <https://doi.org/10.1145/1273496.1273592>. Acesso em: 16 mar. 2023. Citado na p. 36.

RICHARDSON, L. **Beautiful Soup**. 2022. Disponível em: <https://pypi.org/project/beautifulsoup4/>. Acesso em: 10 jan. 2023. Citado na p. 38.

ROCHA., C. A. A.; DIB., M. V. P.; WEIGANG., L.; NUNES., A. F. P.; FARIA., A. V. A.; CAJUEIRO., D. O.; KELY DE MELO., M.; CELESTINO., V. R. R. Using Transfer Learning To Classify Long Unstructured Texts with Small Amounts of Labeled Data. In: INSTICC. PROCEEDINGS of the 18th International Conference on Web Information Systems and Technologies - WEBIST, SciTePress, 2022. P. 201–213. ISBN 978-989-758-613-2. DOI: 10.5220/0011527700003318. Citado nas pp. 18, 27, 66, 73, 74, 78, 82, 90, 102.

RUDER, S.; PETERS, M. E.; SWAYAMDIPTA, S.; WOLF, T. Transfer Learning in Natural Language Processing. In: PROCEEDINGS of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Tutorials. Minneapolis, USA: Association for Computational Linguistics, jun. 2019. P. 15–18. DOI: 10.18653/v1/N19-5004. Disponível em: <https://aclanthology.org/N19-5004>. Acesso em: 16 mar. 2023. Citado nas pp. 36, 53.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed.: Prentice Hall, 2010. Citado nas pp. 44, 45.

SAMMUT, C.; WEBB, G. I. (Ed.). TF–IDF. In: **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. P. 986–987. ISBN 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_832. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_832>. Acesso em: 16 mar. 2023. Citado na p. 42.

SEMBERECKI, P.; MACIEJEWSKI, H. Deep Learning methods for Subject Text Classification of Articles. In: p. 357–360. DOI: 10.15439/2017F414. Citado na p. 33.

SEO, M.; MIN, S.; FARHADI, A.; HAJISHIRZI, H. **Neural Speed Reading via Skim-RNN**. arXiv, 2017. DOI: 10.48550/ARXIV.1711.02085. Disponível em: <https://arxiv.org/abs/1711.02085>. Acesso em: 16 mar. 2023. Citado na p. 32.

SHEN, D.; ZHANG, Y.; HENAO, R.; SU, Q.; CARIN, L. Deconvolutional Latent-Variable Model for Text Sequence Matching. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 32, n. 1, abr. 2018. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/11991>. Acesso em: 16 mar. 2023. Citado na p. 32.

SILVA, B.; ALVES, J.; REBESCHINI, J.; QUEROL, D.; PEREIRA, E.; CELESTINO, V. Data science applied to financial products portfolio. In: ANALS of Meeting of National Association of Post-graduation and Research in Administration. 2021. Citado nas pp. 17, 23, 29, 30, 62, 89, 97, 102.

SUN, C.; QIU, X.; XU, Y.; HUANG, X. How to Fine-Tune BERT for Text Classification? In: SUN, M.; HUANG, X.; JI, H.; LIU, Z.; LIU, Y. (Ed.). **Chinese Computational Linguistics**. Cham: Springer International Publishing, 2019. P. 194–206. ISBN 978-3-030-32381-3. DOI: 10.1007/978-3-030-32381-3{_}16. Citado na p. 35.

THOMPSON, N. C.; GREENEWALD, K.; LEE, K.; MANSO, G. F. **The Computational Limits of Deep Learning**. arXiv, 2020. DOI: 10.48550/ARXIV.2007.05558. Disponível em: <https://arxiv.org/abs/2007.05558>. Acesso em: 16 mar. 2023. Citado na p. 33.

ULIONTSE. **Translators**. pypi.org, 2023. Disponível em: <https://pypi.org/project/translators/>. Acesso em: 16 mar. 2023. Citado na p. 76.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. **Attention Is All You Need**. arXiv, 2017. DOI: 10.48550/ARXIV.1706.03762. Disponível em: <https://arxiv.org/abs/1706.03762>. Acesso em: 16 mar. 2023. Citado nas pp. 50–52, 54.

WANG, G.; LI, C.; WANG, W.; ZHANG, Y.; SHEN, D.; ZHANG, X.; HENAO, R.; CARIN, L. Joint Embedding of Words and Labels for Text Classification. In: PROCEEDINGS of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics,

jul. 2018. P. 2321–2331. DOI: 10.18653/v1/P18-1216. Disponível em: <https://aclanthology.org/P18-1216>. Acesso em: 16 mar. 2023. Citado na p. 50.

WANG, J.; WANG, Z.; ZHANG, D.; YAN, J. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In: p. 2915–2921. DOI: 10.24963/ijcai.2017/406. Citado na p. 33.

WANG, Z.; QU, Z. Research on Web text classification algorithm based on improved CNN and SVM. **2017 IEEE 17th International Conference on Communication Technology (ICCT)**, p. 1958–1961, 2017. Citado na p. 34.

WEIGANG, L. A study of parallel self-organizing map. **arXiv preprint quant-ph/9808025**, 1998. Citado na p. 35.

WEIGANG, L.; SILVA, N. C. da. A study of parallel neural networks. In: IEEE. IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339). 1999. v. 2, p. 1113–1116. Citado na p. 35.

WOLF, T.; DEBUT, L.; SANH, V.; CHAUMOND, J.; DELANGUE, C.; MOI, A.; CISTAC, P.; RAULT, T.; LOUF, R.; FUNTOWICZ, M.; DAVISON, J.; SHLEIFER, S.; PLATEN, P. von; MA, C.; JERNITE, Y.; PLU, J.; XU, C.; LE SCAO, T.; GUGGER, S.; DRAME, M.; LHOEST, Q.; RUSH, A. Transformers: State-of-the-Art Natural Language Processing. In: PROCEEDINGS of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Online: Association for Computational Linguistics, out. 2020. P. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. Disponível em: <https://aclanthology.org/2020.emnlp-demos.6>. Acesso em: 16 mar. 2023. Citado na p. 52.

WU, F.; FAN, A.; BAEVSKI, A.; DAUPHIN, Y. N.; AULI, M. **Pay Less Attention with Lightweight and Dynamic Convolutions**. arXiv, 2019. DOI: 10.48550/ARXIV.1901.10430. Disponível em: <https://arxiv.org/abs/1901.10430>. Acesso em: 16 mar. 2023. Citado na p. 53.

XIAO, L.; WANG, G.; ZUO, Y. Research on Patent Text Classification Based on Word2Vec and LSTM. **2018 11th International Symposium on Computational Intelligence and Design (ISCID)**, v. 01, p. 71–74, 2018. Citado nas pp. 33, 50.

YANG, Z.; YANG, D.; DYER, C.; HE, X.; SMOLA, A.; HOVY, E. Hierarchical Attention Networks for Document Classification. In: PROCEEDINGS of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California: Association for Computational Linguistics, jun. 2016. P. 1480–1489. DOI: 10.18653/v1/N16-1174. Disponível em: <https://aclanthology.org/N16-1174>. Acesso em: 16 mar. 2023. Citado na p. 32.

YOGATAMA, D.; DYER, C.; LING, W.; BLUNSOM, P. **Generative and Discriminative Text Classification with Recurrent Neural Networks**. arXiv, 2017. DOI: 10.48550/ARXIV.1703.01898. Disponível em: <https://arxiv.org/abs/1703.01898>. Acesso em: 16 mar. 2023. Citado na p. 32.

ZELLERS, R.; HOLTZMAN, A.; RASHKIN, H.; BISK, Y.; FARHADI, A.; ROESNER, F.; CHOI, Y. **Defending Against Neural Fake News**. arXiv, 2019. DOI: 10.48550/ARXIV.1905.12616. Disponível em: <https://arxiv.org/abs/1905.12616>. Acesso em: 16 mar. 2023. Citado na p. 69.

ZHANG, H. The Optimality of Naive Bayes. In: v. 2. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004. Citado na p. 45.

ZHANG, L.; LU, L.; NOGUES, I.; SUMMERS, R. M.; LIU, S.; YAO, J. DeepPap: Deep Convolutional Networks for Cervical Cell Classification. **IEEE Journal of Biomedical and Health Informatics**, v. 21, n. 6, p. 1633–1643, 2017. DOI: 10.1109/JBHI.2017.2705583. Citado na p. 32.

ZHAO, K.; HUANG, L.; SONG, R.; SHEN, Q.; XU, H. A Sequential Graph Neural Network for Short Text Classification. **Algorithms**, v. 14, p. 352, dez. 2021. DOI: 10.3390/a14120352. Citado na p. 33.

ZHOU, H. Research of Text Classification Based on TF-IDF and CNN-LSTM. In: JOURNAL of Physics Conference Series. Jan. 2022. v. 2171. (Journal of Physics Conference Series), 012021, p. 012021. DOI: 10.1088/1742-6596/2171/1/012021. Citado na p. 33.

ZHU, Y.; KIROS, R.; ZEMEL, R.; SALAKHUTDINOV, R.; URTASUN, R.; TORRALBA, A.; FIDLER, S. **Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books**. arXiv, 2015. DOI: 10.48550/ARXIV.1506.06724. Disponível em: <https://arxiv.org/abs/1506.06724>. Acesso em: 16 mar. 2023. Citado na p. 69.

# ANEXO A – Artigo WEBIST

# Using Transfer Learning To Classify Long Unstructured Texts with Small Amounts of Labeled Data

Carlos Alberto Alvares Rocha[*,1,8][a], Marcos Vinícius Pinheiro Dib[1,2][b], Li Weigang[*,1,2][c], Andrea Ferreira Portela Nunes[1,3][d], Allan Victor Almeida Faria[1,4][e], Daniel Oliveira Cajueiro[1,5][f], Maísa Kely de Melo[1,6][g] and Victor Rafael Rezende Celestino[1,7][h]

[1]*LAMFO - Lab. of ML in Finance and Organizations, University of Brasilia, Campus Darcy Ribeiro, Brasilia, Brazil*

[2]*TransLab, Department of Computer Science, University of Brasilia, Campus Darcy Ribeiro, Brasilia, Brazil*

[3]*Ministry of Science, Technology and Innovation of Brazil, Federal District, Brazil*

[4]*Department of Statistics, University of Brasília, Federal District, Brazil*

[5]*Department of Economics, University of Brasilia, Federal District, Brazil*

[6]*Department of Mathematics, Instituto Federal de Minas Gerais Campus Formiga, Formiga, Brazil*

[7]*Department of Business Administration, University of Brasilia, Federal District, Brazil*

[8]*PPMEC, Faculty of Technology, University of Brasilia, Federal District, Brazil*

**Emails of corresponding authors: carlosrochacaar@gmail.com, weigang@unb.br*

Keywords: CNN, Deep Learning, MCTI, Longformer, Web Long-text Classification, LSTM, Transfer-learning, Word2vec.

Abstract: Text classification is a traditional problem in Natural Language Processing (NLP). Most of the state-of-the-art implementations require high-quality, voluminous, labeled data. Pre-trained models on large corpora have shown beneficial for text classification and other NLP tasks, but they can only take a limited amount of symbols as input. This is a real case study that explores different machine learning strategies to classify a small amount of long, unstructured, and uneven data to find a proper method with good performance. The collected data includes texts of financing opportunities the international R&D funding organizations provided on their websites. The main goal is to find international R&D funding eligible for Brazilian researchers, sponsored by the Ministry of Science, Technology and Innovation. We use pre-training and word embedding solutions to learn the relationship of the words from other datasets with considerable similarity and larger scale. Then, using the acquired features, based on the available dataset from MCTI, we apply transfer learning plus deep learning models to improve the comprehension of each sentence. Compared to the baseline accuracy rate of 81%, based on the available datasets, and the 85% accuracy rate achieved through a Transformer-based approach, the Word2Vec-based approach improved the accuracy rate to 88%. The research results serve as a successful case of artificial intelligence in a federal government application.

## 1 INTRODUCTION

In Natural Language Processing (NLP), a traditional problem is text classification. It consists in predicting/assigning a predefined category(s) to an input text.

[a] https://orcid.org/0000-0003-0481-6907
[b] https://orcid.org/0000-0003-0998-0597
[c] https://orcid.org/0000-0003-1826-1850
[d] https://orcid.org/0000-0002-3668-8535
[e] https://orcid.org/0000-0002-4300-9334
[f] https://orcid.org/0000-0001-5898-1655
[g] https://orcid.org/0000-0001-8120-9778
[h] https://orcid.org/0000-0001-5913-2997

For this goal, a crucial intermediate task is text representation. Literature covers various neural models for learning text representation, from convolutional (Zhang et al., 2017; Shen et al., 2018) and recurrent models (Yogatama et al., 2017; Seo et al., 2017) to attention mechanisms (Yang et al., 2016; Lin et al., 2017). Alternatively, pre-trained models on large corpora have shown beneficial for text classification and other NLP tasks, possibly preventing the need to train a new model from scratch. One type of pre-trained model is word embeddings, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Another option is contextualized word embeddings, such as CoVe (McCann et al., 2017) and ELMo

(Peters et al., 2018).

These word embeddings contextualized or not, often function as additional features to aid the main task. More recent studies have shown pre-trained language models to be effective in learning common language representations by utilizing a large amount of unlabeled data, such as Generative Pre-trained Transformer - GPT (Radford et al., 2018; Brown et al., 2020)and Bidirectional Encoder Representations from Transformers - BERT (Devlin et al., 2018).

Most of these models go through the training process using a finite set of data, and exceptional results depend on a volume of high-quality, labeled data. The possible variations in quantity or quality of the data can influence the results. In real applications, these models can yield unexpected and unsatisfactory results, affecting the robustness of the solution (Gron, 2017).

The data used to train these models can be unstructured text, the most widely available information on the internet. As much as they are easy to comprehend by humans, they are a challenging input for machines. Therefore, there is a need to develop algorithms and methods capable of processing large amounts of text for various applications (Allahyari et al., 2017). The data can be obtained through information extraction techniques and data mining (Han and Kamber, 2000). Data extraction from diverse platforms results in a nonuniform dataset with unstructured data, where it is not clear where the most relevant information is.

Although Transformer-based approaches have achieved state-of-the-art results in NLP tasks, they are well-suited to deal with relatively short sequences. These models typically limit inputs to $n = 512$ tokens due to the $O(n^2)$ cost of attention hindering their ability to classify long texts (Ainslie et al., 2020). However, there are many NLP applications built around large blocks of text. An example is topic identification of spoken conversations (Hazen, 2011; Kesiraju et al., 2016; Pappagari et al., 2018) and customer satisfaction prediction of call centers (Chowdhury et al., 2016; Luque et al., 2017; Park and Gates, 2009; Meinzer et al., 2017). In the case of call center conversations, the input can vary from short chats to longer ones involving agents trying to solve complex issues that the customers experience, resulting in calls taking as long as an hour or more. An automatic speech recognition (ASR) system transcribes these calls. Such a transcript can exceed the length of 5000 words. Thus ETC (Ainslie et al., 2020) and Longformer (Beltagy et al., 2020) were proposed and obtained state-of-the-art results balancing performance and memory usage.

Another trend employed in NLP tasks is the use of Transfer Learning techniques that allows using the knowledge of an original domain and provides a means of transferring said knowledge to the destination domain to increase the data coverage. While more commonly found in image processing applications, it was shown to be efficient in NLP applications by allowing the sharing of knowledge like similarity in linguistic representation (Ruder et al., 2019). Transfer learning takes the knowledge previously acquired in the original domain and continues the training with new data.

This paper will focus on a more specific problem, creating a Research Financing Products Portfolio (FPP) outside of the Union budget, supported by the Brazilian Ministry of Science, Technology, and Innovation (MCTI), The problem description and conceptual model of FPP/MCTI are shown in Figure 1. The input data includes the text of financing opportunities offered by many institutions worldwide on their websites, such as scholarships, grants, fellowships, and others. A small part of these data was manually labeled by the ministry staff and used in the supervision and training of the classification model, which achieves the accuracy goal defined by the discrimination of the opportunities that allows research financing for Brazilian projects. Due to the nature of the data collection, it has the following characteristics:

- Most of the data is unstructured and nonuniform.
- Texts with high variance in length, reaching up to 5000 tokens/words.
- A short amount of labeled data.

In this article, we explore different Artificial Intelligence (AI) strategies to classify a small amount of long, unstructured, and uneven data to find the appropriate method with good performance.

As the main contribution of this research, we use pre-training and word embedding solutions to learn the relationship of the words from other datasets with considerable similarity and larger scale. Then, using the acquired features, based on the available dataset from MCTI, we apply transfer learning plus deep learning models to improve the comprehension of each sentence to describe the information of the websites. Compared to the baseline accuracy rate of 86%, based on the available datasets, the proposed Word2Vec-based approach in the classification improves the accuracy rate to 88%. All training and testing were done by using Google Colab Pro.

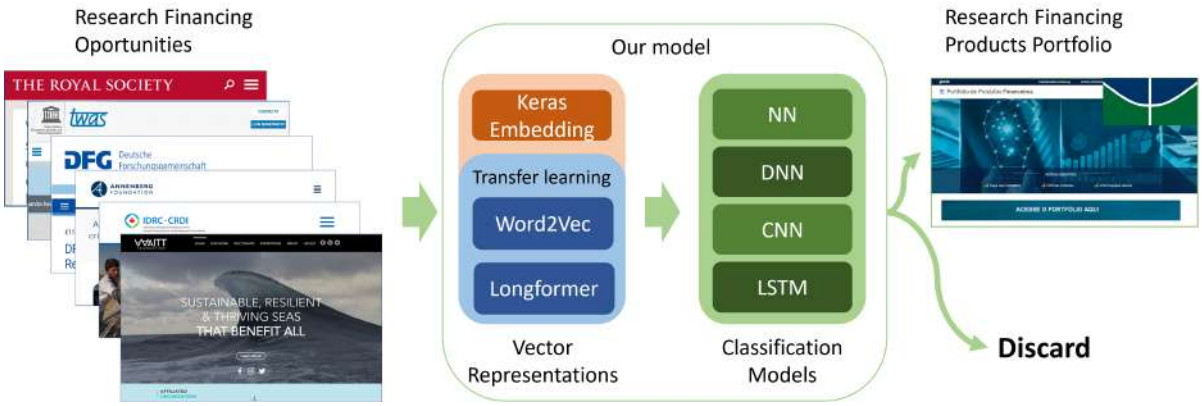



Figure 1: FPP/MCTI classification model.

## 2 PROBLEM DESCRIPTION

Due to recurrent budget cuts in recent years, the MCTI has been looking for ways to develop solutions to finance and promote Science and Technology projects outside the Union budget. A recent partnership was made between MCTI and the University of Brasilia (UnB) to research a semi-automatic system based on AI and data science to assist scholars in finding funding sources for technical projects.

The Research Financing Products Portfolio (FPP) is a system maintained by the Secretariat of Financial Structures and Projects (SEFIP) of MCTI, whose objective is to promote scientific research and raise resources and financing. The FPP presents information on opportunities offered by institutions, foundations, and banks, such as Development Agencies and Multilateral Banks. However, the tools for searching and updating system information are currently manual, making the process long and prone to errors. The expectation is that the system's modernization through Machine Learning (ML) and NLP solutions will provide tools for searching and updating data in an automatic and optimized way.

The research aims to determine and develop intelligent and optimized tools to search, process, organize and present data to compose the FPP. It should also implement a solution with user recommendation and interaction so that the experience allows the continuous improvement of the resources' usability using reinforcement learning.

### 2.1 Main Steps of the Project

The implementation of the project is described in the following steps: Data Scraping, Classification, Summarization, and Recommendation.

**Data Scraping:** The first step in automating identifying and evaluating these sources is reading or capturing the data. For this, a technique called data scraping will be used, which will consist of automated access to leading online platforms that provide financing resources for research projects. The idea is to be able to scan data from these websites and collect information regarding opportunities so that they can be used in the subsequent stages of the project.

**Classification:** With data adequately collected and organized, we can use machine learning techniques to classify opportunities to select those open to Brazilian projects. Since the classification step uses data in text format, it will be necessary to use natural language processing techniques so that it is possible to extract contextual information and allow text interpretation by the computer.

**Summarization:** The next step is data summarization, which uses machine learning and natural language processing techniques to summarize texts of selected and classified opportunities. The summarized texts will facilitate the subsequent dissemination of opportunities.

**Reinforcement Learning and Recommendation:** In order to allow the continuous improvement of the proposed solution, the last stage of the project involves the development of an initial prototype of a recommendation system that uses machine learning. The implementation of the recommendation engine will use reinforcement learning, AI techniques, and feedback regarding the users' experience.

### 2.2 Challenges in the Development of the Project

Artificial intelligence, specifically machine learning and deep learning, has come a long way in the last few

decades (Parloff, 2016). The main reason for this evolution is improved computer performance that made previously impossible tasks possible. Another significant factor was the availability on the internet of large masses of data to train specific-purpose models with an ever-increasing number of parameters.

In the context of this project, we can understand that most of the difficulties associated with the solution are related to these technologies' short time of existence. The work will be developed based on disruptive discoveries and results presented in the last five years and can be currently stated as state-of-the-art. When working with such recent research, it is possible to find problems with replication and implementation with different types of data and different architectures.

The first challenge is related to data acquisition in the scraping stage. It involves scanning several online platforms with different structures, so specific scraping codes are required for each platform. A structural update to a platform can make the scraping code obsolete and deliver incorrect or incomplete data.

Another foreseen challenge concerns the quality of the data obtained through scraping. Due to differences in the text's format, language, and writing, from each scraped platform, it will be difficult to guarantee a standardization of the input data for the training steps in machine learning, which can cause a drop in performance or a worsening of results.

The third challenge is obtaining extra-contextual information currently used by civil servants to identify opportunities. Because of its specificity, this information is difficult to share. The historical context of the platform, and different terminology used by the platform, are good examples of this.

In this paper, we focus on developing Machine Learning solutions with a pre-training strategy for the classification problem.

## 2.3 Input Data

Our work begins with the data obtained from scrapping techniques, and it is vital to show the format and current state of the initial inputted data. The data used was from over 30 different platforms e.g. The Royal Society, Annenberg foundation, and contained 357 rows, but only 260 of them were labeled as shown in Figure 2. Of the data gathered, we were only interested in the main text content and did not use information related to the website URL, title of the page, and other metadata. The content text averages 800 tokens in length, but it has a high variance, reaching up to 5000 tokens as shown in Figure 3.

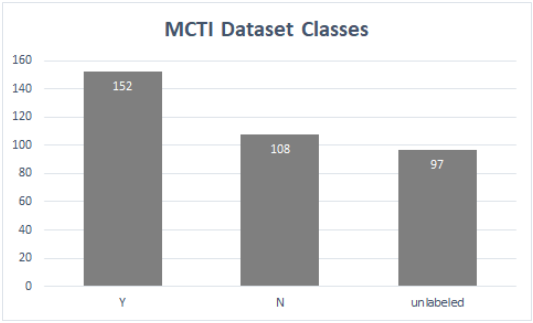



Figure 2: MCTI dataset classes.

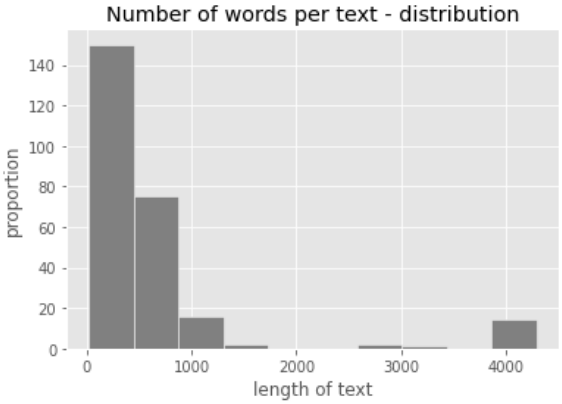



Figure 3: MCTI text length distribution.

# 3 RELATED WORK

In this section, we overview other works related to the classification task in NLP, presenting state-of-the-art works in classification, approaches to long texts, small amounts of labeled data, and transfer-learning.

**Deep Learning Models.** Recent advances in computational power and availability of data granted Deep Learning (DL) a new resurgence (Thompson et al., 2020), more specifically, the progress of artificial Neural Networks (NN), such as Convolutional Neural Networks (CNN) and Long short-term memory (LSTM). These deep neural network models have been used successfully in text and document classification tasks (Minaee et al., 2021).

A DNN is a Deep Neural Network with more hidden layers. These multiple layers of abstraction seem likely to give deep networks a compelling advantage in learning to solve complex pattern recognition problems (Nielsen, 2015). DNNs are composed of connected layers where each layer receives connections from the previous layer and provides connections to the following (Kowsari et al., 2017). The implementa-

tion uses a standard backpropagation algorithm. The output layer has one node for each class to be classified, only one for binary classification, and it is a softmax function. The input needs to be vectorized text using techniques such as word embedding, and the purpose of the network is to learn the relationship between inputs and target spaces using hidden layers (Kowsari et al., 2019).

LSTM stands for Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) and is a neural network based on Recurrent Neural Networks (RNN) with the ability to handle the preservation of short-term memories, specifically circumventing the problem of the vanishing gradient (Pascanu et al., 2013) present in the RNN. It is made of units composed of a cell, an input gate, an output gate, and a forget gate (Gers et al., 2000). Like the RNN, the LSTM contains a chain architecture that considers information from previous nodes, with the difference that it also contains several ports that regulate the amount of information allowed in each node state. This type of architecture associated with word-embeddings has attained excellent results in text classification (Wang et al., 2018; Xiao et al., 2018).

A CNN is a type of Deep Neural Network that applies convolution operations to the input data before feeding it to the NN section. A basic CNN consists of a Convolutional layer, a Pooling layer, and a Fully-connected (FC) or Dense layer. The convolutional layer is responsible for massive data processing using filters and resource maps. The pooling layer is responsible for grouping data to reduce computational complexity and reduce the number of network outputs so that essential characteristics are preserved. The fully connected layer performs the classification task based on features extracted from previous layers (IBM Cloud Education, 2020). While it is most commonly known for image processing (2D) and computer vision (2D to 3D) applications, 1-dimensional convolution exists and has achieved great results in text classification (Kim, 2014).

**Transfer Learning.** The usage of transfer learning in NLP tasks is not new (Pan et al., 2011; Do and Ng, 2005) and has achieved excellent results over the years. The main idea is to transfer knowledge from different source domains to a target domain. A common approach for this is using word vector representations (Ruder et al., 2019; Raina et al., 2007) such as word-embeddings. It is best used when sufficient training data is only available in another domain of interest. In this case, the knowledge transfer could significantly improve learning performance by avoiding expensive data-labeling efforts (Pan and Yang, 2010).

**Pre-training and Fine-tuning.** Within the NLP realm, it has become common to approach problems through transfer learning instead of building a model from scratch by using a model pre-trained on another task and fine-tuning it via further training in the dataset of interest (Mohammed and Ali, 2021; Church et al., 2021). In order to classify texts available on an overload of datasets, it was proposed to use Pre-trained language models (PLM). PLMs are previously trained on a large corpus of text and have some ability to understand text context, such as Transformer-based models (Vaswani et al., 2017).

BERT (Devlin et al., 2018) is still presented as state-of-the-art for most NLP tasks (van den Bulk et al., 2022). Its training is done by conditioning both left and right contexts, simultaneously optimizing for tasks of a masked word and next sentence prediction. BERT-base has an encoder with twelve transformer blocks, twelve self-attention heads, a hidden size of 768, and a maximum input sequence of 512 tokens. In order to perform the classification task, a classification head is included with a simple softmax classifier to return labels' probabilities (Sun et al., 2019). However, this neural network usually only performs when broad information is available in the dataset (Kontonatsios et al., 2020; van Dinter et al., 2021).

In order to process long documents, BERT truncates the text into the max input size (1024 for BERT-large). Another approach presented by (Sun et al., 2019) uses chunking and text fractions to use BERT for long texts. The Longformer (Beltagy et al., 2020) approach uses an attention pattern that combines local and global information while also scaling linearly with the sequence length. It can perform a wide range of document-level NLP tasks without chunking/shortening the long input and without complex architecture to combine information across these chunks achieving state-of-the-art results on the character-level language modeling task. Similarly, ETC (Ainslie et al., 2020) also uses an attention mechanism but differs from the Longformer by combining global-local attention with relative position encodings and flexible masking, enabling it to encode structured inputs similarly as graph neural networks do. Although these implementations performed well when using transformers, they relied on a large number of training data to achieve their results.

Since large-scale machine learning methods or tools such as BERT requires high-performance computing and extremely large-scale data, the tasks faced by our project do not have these resources. Therefore, we must form a technical procedure suitable for this purpose. Based on the existing dataset D1 by MCTI, we find a dataset D2 that has a certain similarity with

D1 and is relatively large in scale. Use D1+D2 data to carry out Pre-training, obtain the corresponding features through Word Embedding, and then fine-tune the DNN models under the guidance of the Transfer learning strategy to achieve Few-shot learning.

**Few-shot Learning.** It is interesting to mention the evolution of the few-example learning. In 1998, the "Once learning" mechanism was proposed for image clustering by one example to simulate human learning behavior (Weigang, 1998) using Self-Organization Map (SOM). This paradigm was also applied to identify the Radar images (Weigang and da Silva, 1999). The researchers (Miller et al., 2000) defined a process to learn from one example through shared densities on transforms. This method used "prior knowledge" to develop a classifier based on only a single training example for each class. Li FeiFei and others (Fei-Fei et al., 2003) developed "One-shot learning" to use knowledge about object categories to classify new objects as humans do. After then, the concept was generalized as "Few-shot learning," accepted by the community, and applied successfully in NLP applications (Brown et al., 2020).

# 4 USING TRADITIONAL DEEP LEARNING METHODS

There have been previous attempts to classify the financing product dataset using non-machine learning approaches. One solution using Naive Bayes and TF-IDF achieved an accuracy of 86% (Silva et al., 2021). Another solution employed keyword search, in which the presence of selected keywords such as Brazil, South America, and others would determine whether it was an eligible opportunity. This solution achieved an accuracy of about 78%. We begin this work by establishing a traditional machine learning model and normalizing the input data, using word embeddings and classification models already consolidated in the NLP community.

## 4.1 Data Normalization

*Normalization* is a preprocessing stage often applied in machine learning systems. The process consists of scaling the data to a needed interval. Also called data scaling, normalization scales the data to study little insights and valuable relationships and to work with the best features of the data (Sree and Bindu, 2018). While not every dataset requires data normalization, when features have different ranges of values, the model might be unable to learn or oscillate back and forth for a long time before finally finding its way to the global/local minimum. Different features with similar ranges of values allow gradient descents to converge more quickly.

Due to the provided data being composed of unstructured and nonuniform texts, normalizing the data became pivotal for defining the development baseline. Normalized data can incorporate more easily into the other layers of the application. The normalization applied was Text Normalization, such as removing HTML special characters and capital letters.

## 4.2 NLP Classification Models

### 4.2.1 Word Embeddings (WE)

The first layer of the proposed model is an embedding layer. Word embedding is a method of extracting features from the data, which can replace one-hot encoding with dimensional reduction. While dealing with textual data, words need to be converted into numbers before being fed into a machine learning model. A simple way is to use one-hot encoding to convert categorical features into numbers. For example, one-hot encoding would transform the input integer values of 0, 1, and 2 to the vectors [1, 0, 0], [0, 1, 0], and [0, 0, 1] respectively (Heaton, 2020). This method results in a dummy feature for each word, which means, for example, 10,000 features for a vocabulary of 10,000 words. This approach is not feasible as it demands large storage space for the word vectors and reduces model efficiency. The embedding layer lets us convert each word into a fixed length vector of defined size. The resultant vector is a dense one with real values instead of just 0's and 1's. In that way, the embedded layer is like a look-up table. The words are the keys in this table, while the dense word vectors are the values.

### 4.2.2 Models

After the embedding, which is just essentially data preprocessing, it is necessary to develop the project further to analyze the input text and classify whether it is a valid research funding opportunity for Brazilian or not. A neural network architecture can be appended to the embedding layer. Various architectures have different performances and training times. For the project, the best option would be chosen empirically upon comparing the results of 4 distinct architectures: Neural Network (NN), Deep Neural Network (DNN), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN). Figure 4 shows the structure of the models.

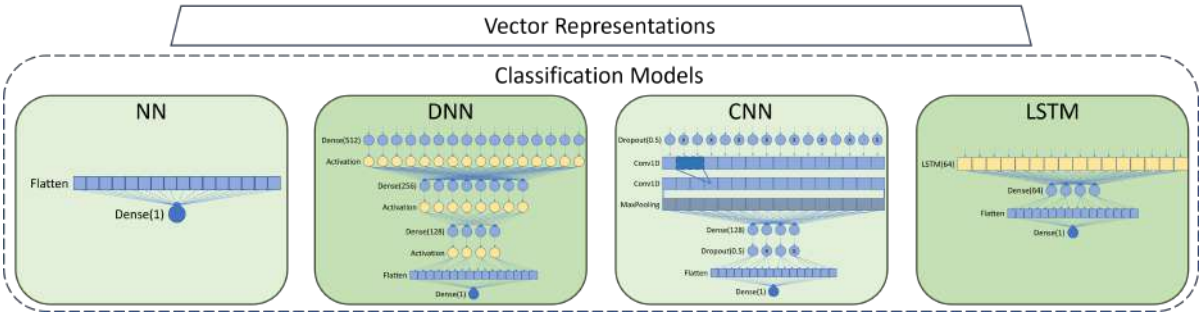


Figure 4: Classification models.

A neural network (NN) here is a simple feedforward neural network with only a single hidden layer, usually called "shallow." Shallow NNs are often limited in the complexity of the problems they can be trained to solve well. Our NN model comprises a simple Flatten layer and a dense classification layer. Our DNN model is composed of a sequence of dense and activation layers varying in size from 512 to 256, to finally 128, and then a Flatten layer feeding into the final dense classification layer. Our CNN model uses a dropout layer feeding into a couple of Conv1D layers and then a MaxPooling layer. After that, we use a hidden layer composed of a dense layer of size 128, followed by another dropout layer, and finally, the Flatten and final dense classification layer. Our LSTM model is formed by an LSTM layer with 64 cells connected to a hidden layer composed of a dense layer of size 64 and, finally, the Flatten and final dense classification layer.

#### 4.2.3 Results from WE + DNN Models

The word embeddings used were single layers of 64 dimensions trained alongside the complete system. We tested the network with architectures of NN, DNN, CNN, and LSTM, and the results with related metrics such as accuracy, precision, recall, and F1 Score are listed in Table 1. When examining the table, it is possible to verify that all the models achieved accuracies already superior to the 78% baseline, reaching values close to 85% in the CNN architecture. Another interesting point to analyze is that this model obtained a precision of 100%, meaning that all opportunities identified as eligible were correctly identified.

Table 1: Results from WE + ML models.

| ML Model | Accuracy | F1 Score | Precision | Recall |
|---|---|---|---|---|
| NN | 0.8269 | 0.8620 | 0.8212 | 0.9129 |
| DNN | 0.8269 | 0.8650 | 0.8952 | 0.8447 |
| CNN | 0.8462 | 0.8756 | 1.0000 | 0.7803 |
| LSTM | 0.8269 | 0.8675 | 0.8276 | 0.9129 |

## 5 USING PRE-TRAINING AND TRANSFER-LEARNING

With the motivation to increase accuracy obtained with baseline implementation, we implemented a transfer learning strategy under the assumption that small data available for training was insufficient for adequate embedding training. In this context, we considered two approaches: i) pre-training word-embeddings using similar datasets for text classification; ii) using transformers and attention mechanisms (Longformer) to create contextualized embeddings. These pre-training approaches can be visualized in Figure 5.

### 5.1 Pre-training Word2Vec Embeddings

Unsupervised training of word-embeddings using the word2vec technique has shown to be very successful in classification tasks (Zhang et al., 2015; Lilleberg et al., 2015).

We need to train word embeddings that can represent the context of the words in our dataset. Since labeled data is scarce, we trained word-embeddings in an unsupervised manner using other datasets that contain most of the words it needs to learn.

#### 5.1.1 Pre-training Datasets

The idea implemented was based on introducing better and better-trained word embeddings in the model. We need larger datasets that contain the words we want to have in vectorial representation in their corpus. For an additional dataset to be applied to improve word-embedding training, it must be compatible with the dataset used to train the classifier. A completely unrelated set of sentences would add, at best, a few examples of the words present in MCTI's different contexts. To evaluate the possible extra datasets, we propose a simple formula:

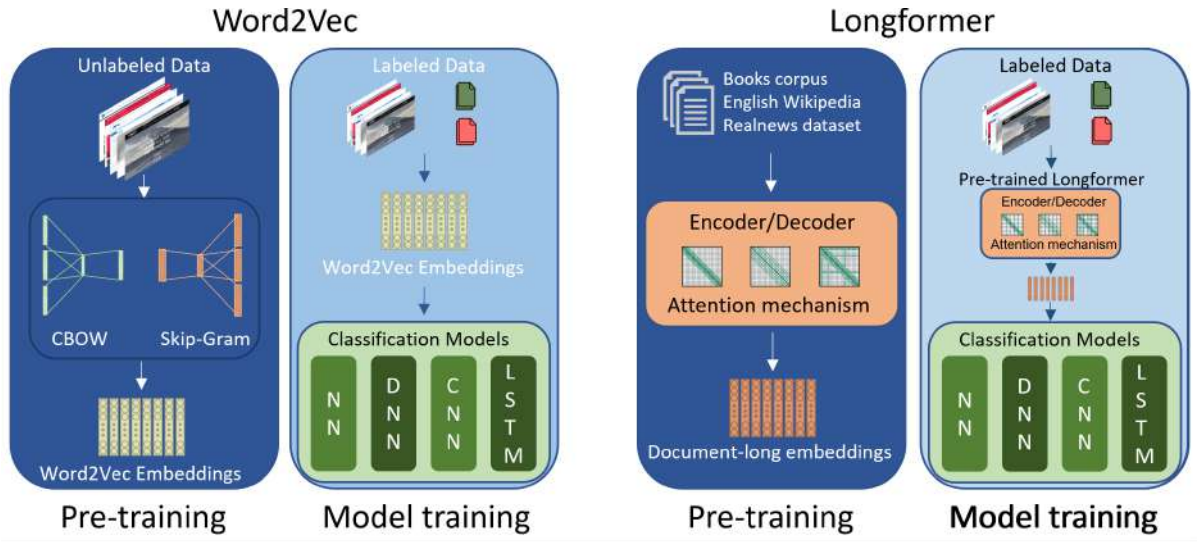



Figure 5: Pre-training models.

$$P_C = \frac{\#(\{Base\} \cap \{New\})}{\#\{Base\}} * 100 \qquad (1)$$

where $\{Base\}$ is the set (unique tokens) of the MCTI dataset, $\{New\}$ is the set of the target additional dataset, and # denotes the number of elements in a Set. This formula calculates the percentage of the words in the original dataset present in the new dataset. The maximum number is 100 (all tokens are present in the new dataset).

The equation (1) yields a percentage of similarity that will help assess which datasets to use for the pre-training. This determination is essential once we do not have the computational power to pre-train with massive data like professional solutions such as BERT and GPT (Weigang et al., 2022).

We searched for datasets from the Kaggle, a platform with over a thousand available NLP datasets, and the closest we found was the BBC News Articles dataset. After applying the compatibility by equation (1), only approximately 57% of the words needed were presented, which we considered not high enough.

The alternative was to use web scraping algorithms to acquire more unlabeled data from the same sources, which would give a higher chance of providing compatible texts. The original dataset had 357 entries, with 260 of them labeled. We obtained 518 new data elements with the second scraping. When comparing the 260 original elements to the 518 new unlabeled ones, we achieved over 75% compatibility, indicating the alternative assumption was correct. Table 2 displays the result of the comparisons made. Compatibility means the percentage of the original dataset that is contained in the new dataset.

Table 2: Compatibility results (*base = labeled MCTI dataset entries).

| Dataset | Compatibility to base* |
|---|---|
| Labeled MCTI | 100% |
| Full MCTI | 100% |
| BBC News Articles | 56.77% |
| New unlabeled MCTI | 75.26% |

Figure 6 shows a representation of the weights trained by Word2Vec. It is obtained through dimensionality reduction using t-Distributed Stochastic Neighbor Embedding (t-SNE). In *a)*, the training was done using the 260 labeled data; in *b)*, it was used 875 unlabeled data. In the image, it is possible to observe that the concentrated/dense area increased considerably with the addition of the new dataset and its tokens, demonstrating the enrichment of the weights concerning the desired contextual information. If pre-training was done with an incompatible dataset, it would be likely that other large dense areas would appear and probably interfere with the meaning of the words.

## 5.2 Model Training with Word2Vec Embeddings

Now we have a pre-trained model of word2vec embeddings that has already learned relevant meanings for our classification problem. We can couple it to our classification models (Fig. 4), realizing transfer-learning and then training the model with the labeled data in a supervised manner. The new coupled model can be seen in Figure 5 under word2vec model training.

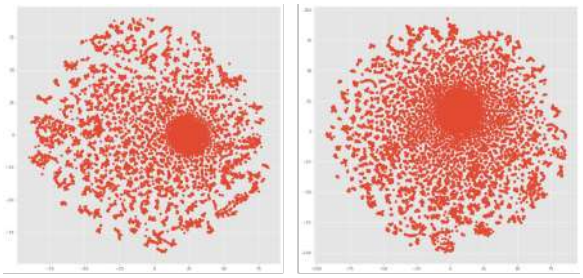



Figure 6: Pre-trained weights by Word2Vec: a) weights trained with labeled MCTI data; b) weights trained with Full MCTI + New unlabeled MCTI data.

The Table 3 shows the obtained results with related metrics. With this implementation, we achieved new levels of accuracy with 86% for the CNN architecture and 88% for the LSTM architecture.

Table 3: Results from Pre-trained WE + ML models.

| ML Model | Accuracy | F1 Score | Precision | Recall |
|---|---|---|---|---|
| NN | 0.8269 | 0.8545 | 0.8392 | 0.8712 |
| DNN | 0.7115 | 0.7794 | 0.7255 | 0.8485 |
| CNN | 0.8654 | 0.9083 | 08486 | 0.9773 |
| LSTM | 0.8846 | 0.9139 | 0.9056 | 0.9318 |

## 5.3 Transformer-based Implementation

Another way we used pre-trained vector representations was by use of a Longformer (Beltagy et al., 2020). We chose it because of the limitation of the first generation of transformers and BERT-based architectures involving the size of the sentences: the maximum of 512 tokens. The reason behind that limitation is that the self-attention mechanism scale quadratically with the input sequence length $O(n^2)$ (Beltagy et al., 2020). The Longformer allowed the processing sequences of a thousand characters without facing the memory bottleneck of BERT-like architectures and achieved SOTA in several benchmarks.

For our text length distribution in Figure 3, if we used a Bert-based architecture with a maximum length of 512, 99 sentences would have to be truncated and probably miss some critical information. By comparison, with the Longformer, with a maximum length of 4096, only eight sentences will have their information shortened.

To apply the Longformer, we used the pre-trained base (available on the link) that was previously trained with a combination of vast datasets as input to the model, as shown in figure 5 under Longformer model training. After coupling to our classification models, we realized supervised training of the whole model. At this point, only transfer learning was applied since more computational power was needed to realize the fine-tuning of the weights. The results with related metrics can be viewed in table 4. This approach achieved adequate accuracy scores, above 82% in all implementation architectures.

Table 4: Results from Pre-trained Longformer + ML models.

| ML Model | Accuracy | F1 Score | Precision | Recall |
|---|---|---|---|---|
| NN | 0.8269 | 0.8754 | 0.7950 | 0.9773 |
| DNN | 0.8462 | 0.8776 | 0.8474 | 0.9123 |
| CNN | 0.8462 | 0.8776 | 0.8474 | 0.9123 |
| LSTM | 0.8269 | 0.8801 | 0.8571 | 0.9091 |

# 6 DISCUSSION

In this section, we will discuss the problem, the models used and the results obtained in the context of classification in NLP. Table 5 shows the overview of the final results obtained by the experiments:

Table 5: Comparison of the results of three scenarios.

| Model | Accuracy | F1 Score | Precision | Recall |
|---|---|---|---|---|
| Baseline | | | | |
| Bag of words | 0.86 | 0.85 | 0.85 | 0.85 |
| Key words | 0.7808 | 0.7802 | 0.9358 | 0.6711 |
| Keras Word-embedding | | | | |
| NN | 0.8269 | 0.8620 | 0.8212 | 0.9129 |
| DNN | 0.8269 | 0.8650 | 0.8952 | 0.8447 |
| CNN | 0.8462 | 0.8756 | 1.0000 | 0.7803 |
| LSTM | 0.8269 | 0.8675 | 0.8276 | 0.9129 |
| Pre-trained Word2Vec Embeddings | | | | |
| NN | 0.8269 | 0.8545 | 0.8392 | 0.8712 |
| DNN | 0.7115 | 0.7794 | 0.7255 | 0.8485 |
| CNN | 0.8654 | 0.9083 | 08486 | 0.9773 |
| LSTM | 0.8846 | 0.9139 | 0.9056 | 0.9318 |
| Pre-trained Longformer | | | | |
| NN | 0.8269 | 0.8754 | 0.7950 | 0.9773 |
| DNN | 0.8462 | 0.8776 | 0.8474 | 0.9123 |
| CNN | 0.8462 | 0.8776 | 0.8474 | 0.9123 |
| LSTM | 0.8269 | 0.8801 | 0.8571 | 0.9091 |

When analyzing the data, we can see an aspect that hinders the visualization of the accuracy improvements: the values changes in 1.9% jumps. The amount of labeled testing data is minimal, and since we carry out 80-20 training segmentation, we only verify 52 items, which explains the jumps.

It is possible to verify that most the tested classification models showed improved accuracy after pre-training. In particular, the network composed of pre-trained word2vec embeddings + LSTM network obtained the best result for the target dataset of 88% accuracy and over 90% precision.

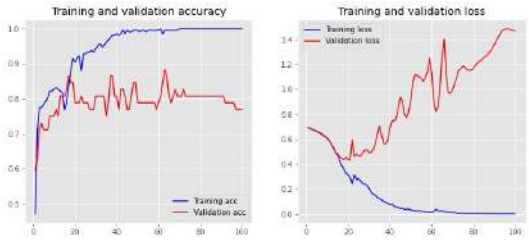


Figure 7: LSTM model with Word2Vec training curve.

Taking a deeper look at the W2V + LSTM training curve in Figure 7, we can see that the model reached its best accuracy around the 60th epoch. We can also see constant fluctuations in the accuracy values, which are very common in LSTM models. After that, even though the training accuracy kept improving, there was a pattern of overfitting where the validation accuracy declined.

The Longformer results did surpass the simple embeddings model and, although satisfactory, can probably be improved by fine-tuning the model. That will require more computing power and optimizations and will be done in further studies.

# 7 CONCLUSION AND FUTURE WORK

Advances in artificial intelligence in recent years have aroused the interest of MCTI in automating the process of identifying and selecting relevant opportunities for Brazilian projects. This task plays a significant role in helping to promote scientific research and in its funding.

This research has developed machine learning-based methods to automatically process the collected text for the Financing Products Portfolio (FPP) organized by the Brazilian Ministry of Science, Technology, and Innovation (MCTI). The main objective consists of performing automated searches on known digital platforms using data scraping techniques to collect information about opportunities. Then, these data will serve as input for an algorithm that uses NLP methods to classify opportunities in terms of eligibility for Brazilian projects.

In this article, we reported the following achievements on the first two steps of the project, mentioned in section 2.1:

- presentation of the collected data, including the texts of financing opportunities provided by many R&D funding organizations worldwide.
- presentation of the performance by the previous study, which established the accuracy rate of 86% in the text classification as the baseline.
- the achievement of the accuracy rate of 84% by the solution using Word Embedding and Deep learning such as NN, DNN, CNN, and LSTM.
- the achievement of the accuracy rate of 88% by the solution of pre-training Word2Vec embeddings with transfer learning plus deep learning models such as NN, DNN, CNN, and LSTM.
- the achievement of the accuracy rate of 84% with the pre-trained Longformer plus deep learning models such as NN, DNN, CNN, and LSTM.

All the data and models used here, as well as the results obtained, are publicly available in github[1].

It is worth mentioning the comparative contribution of this research. It is well-known that Google, Microsoft, IBM, OpenAi, and other prominent international AI companies have human, data, and computing resources that allowed them to develop well-known machine learning methods and theories such as Transformer, GPT, BERT, and Vision Transformer. In our case, however, we are faced with limited resources and small-scale structured data. We introduced the pre-learning strategy to learn the features from larger-scale unstructured data and then used transfer learning methods to improve the accuracy of data classification through deep learning. Achieving the same goal with limited resources is an important initiative and is an essential measure of technical progress in developing countries, especially in the development of AI technology.

Apart from the benefit of automating the process of identifying and selecting opportunities that already exist today (which is the focus of this research), we can also emphasize that the technology developed here may be applied in several fields of public management for other related sectors from federal or state government. In particular, the proposed solution can benefit other research projects in developing countries, which face similar challenges with limited resources and small-scale structured data.

# ACKNOWLEDGEMENTS

The Brazilian Ministry of Science, Technology, and Innovation (MCTI) has partially supported this project. We sincerely thank Dr. Joao Gabriel Souza, who leads the dataset construction and kindly shared the data for this study.

[1] https://github.com/chap0lin/WEBIST2022

# REFERENCES


Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. (2020). Etc: Encoding long and structured inputs in transformers.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chowdhury, S. A., Stepanov, E. A., and Riccardi, G. (2016). Predicting User Satisfaction from Turn-Taking in Spoken Conversations. In *Proc. Interspeech 2016*, pages 2910–2914.

Church, K. W., Chen, Z., and Ma, Y. (2021). Emerging trends: A gentle introduction to fine-tuning. *Natural Language Engineering*, 27(6):763–778.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Do, C. B. and Ng, A. Y. (2005). Transfer learning for text classification. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Fei-Fei, L., Fergus, R., and Perona, P. (2003). A bayesian approach to unsupervised one-shot learning of object categories. In *proceedings ninth IEEE international conference on computer vision*, pages 1134–1141. IEEE.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

Gron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition.

Han, J. and Kamber, M. (2000). Data mining: Concepts and techniques.

Hazen, T. J. (2011). Mce training techniques for topic identification of spoken audio documents. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2451–2460.

Heaton, J. (2020). Applications of deep neural networks. *arXiv preprint arXiv:2009.05673*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

IBM Cloud Education (2020). What are convolutional neural networks? https://www.ibm.com/cloud/learn/convolutional-neural-networks. IBM Cloud Learn Hub.

Kesiraju, S., Burget, L., Szőke, I., and Černocký, J. (2016). Learning document representations using subspace multinomial model. In *Proceedings of Interspeech 2016*, pages 700–704. International Speech Communication Association.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Kontonatsios, G., Spencer, S., Matthew, P., and Korkontzelos, I. (2020). Using a neural network-based feature extraction method to facilitate citation screening for systematic reviews. *Expert Systems with Applications: X*, 6:100030.

Kowsari, Meimandi, J., Heidarysafa, Mendu, Barnes, and Brown (2019). Text classification algorithms: A survey. *Information*, 10(4):150.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., and Barnes, L. E. (2017). HDLTex: Hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE.

Lilleberg, J., Zhu, Y., and Zhang, Y. (2015). Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pages 136–140.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding.

Luque, J., Segura, C., Sánchez, A., Umbert, M., and Galindo, L. A. (2017). The role of linguistic and prosodic cues on the prediction of self-reported satisfaction in contact centre phone calls. In *Proc. Interspeech 2017*, pages 2346–2350.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. In *NIPS*.

Meinzer, S., Jensen, U., Thamm, A., Hornegger, J., and Eskofier, B. (2017). Can machine learning techniques predict customer dissatisfaction? a feasibility study for the automotive industry. *Artif. Intell. Res.*, 6:80–90.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, USA. Curran Associates Inc.

Miller, E. G., Matsakis, N. E., and Viola, P. A. (2000). Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., and Gao, J. (2021). Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3).

Mohammed, A. H. and Ali, A. H. (2021). Survey of BERT (Bidirectional Encoder Representation Transformer)

types. In *Journal of Physics: Conference Series*, volume 1963, page 012173. IOP Publishing.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination press San Francisco, USA.

Pan, S. J., Tsang, I. W.-H., Kwok, J. T.-Y., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22:199–210.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pappagari, R., Villalba, J., and Dehak, N. (2018). Joint verification-identification in end-to-end multi-scale cnn framework for topic identification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203.

Park, Y. and Gates, S. C. (2009). Towards real-time measurement of customer satisfaction using automatically generated call transcripts. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 1387–1396, New York, USA. Association for Computing Machinery.

Parloff, R. (2016). Why deep learning is suddenly changing your life. *Fortune. New York: Time Inc*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Int. Conf. on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th Int. Conf. on Machine Learning*, ICML '07, page 759–766, New York, USA. Association for Computing Machinery.

Ruder, S., Peters, M. E., Swayamdipta, S., and Wolf, T. (2019). Transfer learning in natural language processing. In *Proceedings of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, USA. Association for Computational Linguistics.

Seo, M., Min, S., Farhadi, A., and Hajishirzi, H. (2017). Neural speed reading via skim-rnn.

Shen, D., Zhang, Y., Henao, R., Su, Q., and Carin, L. (2018). Deconvolutional latent-variable model for text sequence matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Silva, B., Alves, J., Rebeschini, J., Querol, D., Pereira, E., and Celestino, V. (2021). Data science applied to financial products portfolio. In *Anals of Meeting of National Association of Post-graduation and Research in Administration*.

Sree, K. and Bindu, C. (2018). Data analytics: Why data normalization. *International Journal of Engineering and Technology (UAE)*, 7:209–213.

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to fine-tune bert for text classification? In Sun, M., Huang, X., Ji, H., Liu, Z., and Liu, Y., editors, *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Thompson, N. C., Greenewald, K., Lee, K., and Manso, G. F. (2020). The computational limits of deep learning.

van den Bulk, L. M., Bouzembrak, Y., Gavai, A., Liu, N., van den Heuvel, L. J., and Marvin, H. J. (2022). Automatic classification of literature in systematic reviews on food safety using machine learning. *Current Research in Food Science*, 5:84–95.

van Dinter, R., Catal, C., and Tekinerdogan, B. (2021). A decision support system for automating document retrieval and citation screening. *Expert Systems with Applications*, 182:115261.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Wang, G., Li, C., Wang, W., Zhang, Y., Shen, D., Zhang, X., Henao, R., and Carin, L. (2018). Joint embedding of words and labels for text classification.

Weigang, L. (1998). A study of parallel self-organizing map. *arXiv preprint quant-ph/9808025*.

Weigang, L. and da Silva, N. C. (1999). A study of parallel neural networks. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 2, pages 1113–1116. IEEE.

Weigang, L., Enamoto, L. M., Li, D. L., and Rocha Filho, G. P. (2022). New directions for artificial intelligence: Human, machine, biological, and quantum intelligence. *Frontiers of Information Technology & Electronic Engineering*, 23(6):984–990.

Xiao, L., Wang, G., and Zuo, Y. (2018). Research on patent text classification based on word2vec and lstm. In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 01, pages 71–74.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Yogatama, D., Dyer, C., Ling, W., and Blunsom, P. (2017). Generative and discriminative text classification with recurrent neural networks.

Zhang, D., Xu, H., Su, Z., and Xu, Y. (2015). Chinese comments sentiment classification based on word2vec and svmperf. *Expert Systems with Applications*, 42(4):1857–1863.

Zhang, L., Lu, L., Nogues, I., Summers, R. M., Liu, S., and Yao, J. (2017). Deeppap: Deep convolutional networks for cervical cell classification. *IEEE Journal of Biomedical and Health Informatics*, 21(6):1633–1643.