Instituto Politécnico de Tomar

MODELO DE CLASSIFICAÇÃO DE DOCUMENTO

Relatório de Projecto de Avaliação

José Nduli Futi Pinto

Mestrado em Analise Inteligencia Organizacional

Tomar, July 20, 2023

RESUMO

Classificação de documentos com aprendizagem profunda é uma abordagem avançada para categorizar automaticamente documentos em diferentes classes ou categorias, como por exemplo os sentimentos, tópicos ou outros tipos de conteúdos. Essa abordagem envolve o uso de redes neurais profundas, que são modelos de aprendizado de máquina capazes de aprender representações complexas dos dados.

Neste relatório apresento uma aplicação baseada em redes neurais profundas (Deep Learning) para a classificação de diferentes tipos de documentos. Inclui todas as técnicas necessárias para resultados eficientes em cada etapa do projecto.

Os documentos são processados para remover ruídos, normalizar o texto (remover pontuações, letras maiúsculas, etc.) e criar representações numéricas dos textos, como vetores de palavras (Word Embeddings) ou representações baseadas em sub-palavras.

Foi escolhida uma arquitetura de rede neural profunda, redes neurais recorrentes (RNNs). O modelo é treinado num conjunto de dados rotulados, onde os rótulos indicam as categorias às quais os documentos pertencem.

Durante o treinamento, o modelo ajusta os seus pesos e parâmetros para minimizar a diferença entre as previsões e os rótulos reais. Isso é feito usando algoritmos de optimização e funções de perda. O modelo treinado é avaliado num conjunto separado de dados de validação ou teste para verificar o seu desempenho e ajustar os hiperparâmetros, se necessário. Esse processo ajuda a evitar o sobreajuste (overfitting) do modelo aos dados de treinamento.

Finalmente, o modelo é avaliado num conjunto de teste independente para medir a sua capacidade de generalização em dados não vistos. Isso fornece uma estimativa realista do desempenho do modelo em ambientes do mundo real. Uma vez que o modelo atinja um desempenho satisfatório, ele pode ser implantado para classificar automaticamente novos documentos em categorias relevantes. Para o nosso caso a perda (Loss): 0.546 indica que o modelo está fazendo previsões razoavelmente próximas aos valores reais, o que é um bom sinal. E a precisão (Accuracy) : 0.860 (ou 86%) sugere que o modelo classificou corretamente 86%. Isso é uma taxa de acerto relativamente alta e indica que o modelo está realizando bem a tarefa.

Palavras-chave: Aplicacões Web, Classificação de Documentos com Deep Learning, LSTM, Análise de textos, Aprendizado Profundo, Aprendizado de Máquina

ABSTRACT

Document classification with deep learning is an advanced approach to automatically categorize documents into different classes or categories, such as sentiments, topics or other types of content. This approach involves the use of deep neural networks, which are machine learning models capable of learning complex representations of data. Deep learning has a variety of applications, including analyzing sentiment on social media, categorizing emails, classifying news articles, automatically triaging legal documents, and more.

In this report I present an application based on deep neural networks (Deep Learning) for classifying different types of documents. Includes all the techniques necessary for efficient results at each stage of the project.

Documents are processed to remove noise, normalize the text (remove punctuations, capital letters, etc.) and create numerical representations of the texts, such as word vectors (Word Embeddings) or representations based on sub-words.

A deep neural network architecture such as recurrent neural networks (RNNs) was chosen for this project. The model is trained on a labeled dataset, where the labels indicate the categories of the documents belong to.

During training, the model adjusts its weights and parameters to minimize the difference between predictions and actual labels. This is done using optimization algorithms and loss functions. And the trained model is evaluated on a separate set of validation or test data to check its performance and tune hyperparameters if necessary. This process helps prevent overfitting the model to the training data.

Finally, the model is evaluated on an independent test set to measure its generalization ability to unseen data. This provides a realistic estimate of the model's performance in real-world environments. Once the model achieves satisfactory performance, it can be deployed to automatically classify new documents into relevant categories. In our case, the loss (Loss): 0.546 indicates that the model is making predictions reasonably close to the real values, which is a good sign. And Accuracy: 0.860 (or 86%) suggests that the model correctly classified 86%. This is a relatively high hit rate and indicates that the model is performing the task well.

Keywords: Web Applications, Document Classification with Deep Learning, LSTM, Text Analysis, Deep Learning, Machine Learning.

AGRADECIMENTOS

Em primeiro lugar queria agradecer a minha família pelo carinho, apoio e paciência que tiveram ao longo destes anos e que sem eles não seria possível chegar ate aqui.

Agradeço ao meu professor e orientador, Carlos Mora por toda a disponibilidade, dedicação o e apoio fundamental para a realizacão deste projecto e sem esquecer de outros professores Sergio Rodrigues e Sandra Jardim que também contribuiram muito no projecto.

Uma palavra de agradecimento especial para a Albertina Lelo e pelo apoio e ajuda que me deu na fase final do projeto e a minha formação no geral.

A todos que indirectamente apoiaram não simplesmente por material más por palavras encorajadouras, motivações e outros durante o percurso do ano letivo.

ÍNDICE

LISTA DE FIGURAS

CAPÍTULO 1

1.1 INTRODUÇÃO

Inteligência Artificial - IA, apesar de não ter uma única definição e o seu conceito ser de difícil definição, tem ganhado espaço a cada ano, se mostrando a grande aposta para as aplicações em automação e em diversas áreas da tecnologia da informação. De maneira genérica, podemos tentar delimitar um conceito para IA como sendo: uma inteligência similar à humana exibida por máquinas, com o objetivo de gerar a interface de interação entre um sistema inteligente e o mundo real. Desde a criação do conceito de IA, muito se avançou em pesquisas e aplicações, porém tem sido cada vez maior o investimento neste campo de estudo.

1.1.1 Comunicação

A comunicação, como uma necessidade básica da condição humana, juntamente com a existência da Web permitem que uma vasta quantidade de textos escritos e falados seja gerada diariamente. Dado o conteúdo textual presente em redes sociais, aplicativos de bate-papo, e-mails, análises de produtos, artigos de notícias, trabalhos de pesquisa e ebooks, tornou-se vital a existência de um processamento automático de textos a fim de oferecer assistência ou tomar decisões para diversas tarefas diárias.

1.1.2 Linguagem Natural

A capacidade de entender textos ou áudios em linguagem natural por uma máquina é um problema que vem sendo investigado há muito tempo [Chollet 2021]. As primeiras tentativas de construção de sistemas de Processamento de Linguagem Natural (PLN, de Natural Language Processing ou NLP) foram feitas através da análise intrínseca de linguagens que são naturalmente moldadas por um processo de evolução (por isso o termo“natural”). O PLN moderno envolve não apenas a habilidade de entendimento de uma linguagem como também possibilita, de forma automática, a extração de informações por meio de tarefas, tais como Classificação de Textos, Reconhecimento de Entidades Nomeadas (NER, de Named Entity Recognition), Desambiguação do Sentido das Palavras, e Part-of-Speech (POS) tagging.

1.1.3 Modelo de Aprendizagem

Modelos de Aprendizagem Profunda, do inglês Deep Learning (também conhecida como Aprendizado Profundo ou Redes Neurais Profundas), aprendem vários níveis de representação de complexidade/abstração dos dados de forma crescente. Vários fatores evidenciam porque esses modelos têm sido amplamente usados em tarefas de PLN:(i) exigem pouca engenharia de features; (ii) produzem representações vetoriais que capturam similaridades de unidades linguísticas (palavras, por exemplo) presentes em textos, permitindo que sistemas de PLN possuam uma espécie de dependência de conhecimento;(iii) permitem aprendizado não supervisionado ou semi-supervisionado, o que é importante quando se tem um grande volume de dados e nenhum rótulo; (iv) aprendem vários níveis de representação, possibilitando que o nível mais baixo geralmente possa ser compartilhado entre diferentes tarefas; e (v) naturalmente lidam com a recursividade da linguagem humana, sendo capazes de capturar informações de forma sequencial.

Os algoritmos modernos de PLN baseiam-se na aprendizagem mecânica, especialmente na aprendizagem de máquinas estatísticas. O paradigma da aprendizagem mecânica é diferente do da maioria das tentativas anteriores de processamento da linguagem. Anteriormente, implementações de tarefas de processamento de linguagem envolviam a codificação direta de grandes conjuntos de regras. O paradigma da aprendizagem automática (ou aprendizagem automática) induz a aprendizagem automática de regras através de análises de corpora de exemplos típicos do mundo real ao invés de usar algoritmos gerais de aprendizagem (muitas vezes, embora nem sempre, baseados em inferência estatística). Um corpus (plural "corpora") é um conjunto de documentos (ou frases individuais) que foram anotados à mão com os valores corretos a serem aprendidos.

1.2 OBJECTIVO

A criação de modelo de aprendizagem de processamento de documentos é automatizar a análise, compreensão e extracção de informações úteis a partir de documentos de texto, como artigos, relatórios, e-mails, contratos, entre outros. Esses modelos são uma parte essencial da área de Processamento de Linguagem Natural (PLN) e têm uma variedade de aplicações práticas em diferentes sectores, incluindo negócios, saúde, jurídico, academia e muito mais.

Em resumo, os modelos de aprendizagem de processamento de documentos têm como objectivo aumentar a eficiência e a precisão na análise de documentos de texto, possibilitando a extracção de informações valiosas e a construção de sistemas inteligentes capazes de lidar com uma ampla variedade de tarefas relacionadas ao processamento de linguagem natural.

1.2.1 Organização do Documento

Este relatório documentado define e descreve conceitos sobre modelo de aprendizagem de processamento de documentos através de uma aplicação web que tem como objectivo aumentar a eficiência e a precisão na análise de documentos de texto, possibilitando a extracção de informações valiosas e a construção de sistemas inteligentes capazes de lidar com uma ampla variedade de tarefas relacionadas ao processamento de linguagem natural.

CAPÍTULO 2

2.1 ANÁLISE

Neste capítulo será descrito os requisitos necessários para a implementação deste projecto de dissertação. Para o efeito serão listados os requisitos funcionais e não funcionais

2.2.1 Requisitos Funcionais

Os requisitos funcionais focam-se na funcionalidade e os serviços do sistema, ou seja, nas funções que o sistema deve fornecer ao utilizador e como o sistema se deve comportar para garantir essas funcionalidades.

Para alcançar cada objectivo referido na secção 2 foram definidos requisitos funcionais do sistema. Estes encontram-se enumerados de seguida:

- Visualizar a lista de possibilidades de introduzir texto a ser classificados:
    - Por ficheiros no formato txt;
    - Por textos;
    - Por ficheiro no formato pdf.
- Validar ou classificar o texto introduzido;
- Visualizar estatísticas ou o resultado sobre o texto introduzido;
- Capaz de treinar novos modelos inserindo apenas o dataset.

2.2.2 Requisitos não Funcionais

Os requisitos não funcionais referem-se a critérios que podem ser usados para avaliar as operações de um sistema, em vez de comportamentos específicos. Os requisitos não-funcionais para este projecto, são os seguintes:

- **Usabilidade**: o interface da aplicação deverá ser intuitiva, fácil de usar, e de aprender a utilizar as funcionalidades que a aplicação suporta.
- **Fiabilidade**: a aplicação deve apresentar o mínimo de falhas possíveis na sua utilização.

- **Disponibilidade**: a aplicação deverá ter alta disponibilidade nas funcionalidades que apresenta.
- **Portabilidade**: a aplicação deverá estar disponível em qualquer dispositivo, tanto desktop como mobile.
- **Manutenção**: a aplicação deverá ser de fácil manutenção, permitindo uma fácil actualização. Deverá permitir uma fácil adaptação a novos requisitos.

2.2.3 Caso de Uso

O diagrama de caso de uso descreve a funcionalidade proposta para um novo sistema que será projectado, sendo esta uma excelente ferramenta para representar todas as interacções possíveis, entre o utilizador e o sistema. Estes diagramas são usados para descrever a sequência de eventos que um actor que usa a aplicação deverá seguir para completar um processo.

Sistema - o sistema representa, neste caso, a aplicação *Web Data Captures*.

Actor - o actor especifica o papel executado por um utilizador ou outro sistema que interage com o sistema. Este deve ser externo ao sistema, deve ter associações exclusivamente para casos de uso, componentes ou classes e e representado por um boneco (stick man).

No caso específico deste relatório, o actor terá o papel de utilizador que irá interagir com o sistema.

Caso de uso - é uma especificação de um conjunto de acções executadas por um sistema, que contém um resultado observável. É representado por uma elipse, com o nome deste dentro ou abaixo. Num diagrama de casos de uso, os casos de uso estão associados ao actor que os executa. Com base nos requisitos funcionais levantados na secção 3.1, foram definidos os seguintes caso de uso ilustrados na figura 3.1.

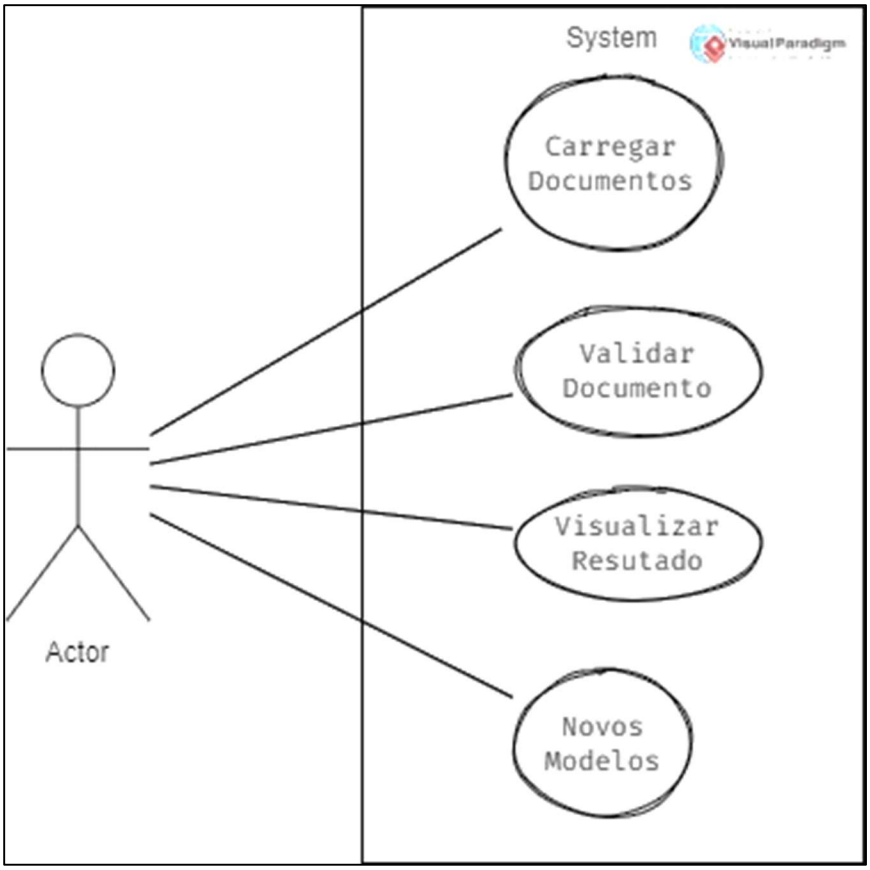


Figura 1 - Diagrama de Casos de Uso da aplicação Web

Para cada caso de uso mencionado na figura 1 são descritos, nas subsecções seguintes, os objectivos, atores, fluxo de acções, o resultado previsto e para alguns casos o estado de erro associado.

2.2.3.1 Visualizar Campos de introdução de texto*:*

- **Objectivos**: capaz de inserir o texto ou documento para a sua classificação
- **Actor**: utilizador.
- **Fluxo de acções**: aceder a aplicação.
- **Resultado previsto**: o utilizador conseguir ver diferentes tipos forma de inserção de textos

2.2.3.2 Validação do texto Introduzido:

- **Objectivos**: capaz de carregar o botão para o sistema fazer a leitura e classificar o texto
- **Actor**: utilizador.
- **Fluxo de acções**: aceder a aplicação.

- **Resultado previsto**: o utilizador conseguir ver o botão para validar e apôs validar capaz de ver o mesmo a processar o resultado.

2.2.3.3 Visualizar Resultado:

- **Objectivos** mostrar a respota a da classificação no ecrã
- **Actor**: sistema.
- **Fluxo de acções**: aceder a aplicação.
- **Resultado previsto**: o utilizador conseguir ver o o resultado da classificação feita pelo sistema de modelo de classificação de texto.

2.2.3.4 Treinamento de modelo:

- **Objectivos** capaz de treinar novos modelo de classificação de documento
- **Actor**: utilizador.
- **Fluxo de acções**: aceder a aplicação.
- **Resultado previsto**: o utilizador conseguir inserir um novo dataset para criar novos modelos de classificação de documentos.

CAPÍTULO 3

3.1 IMPLEMENTAÇÃO

Neste capítulo será apresentado uma descrição o do processo de implementação do projecto, começando com uma apresentação da tecnologia usada, seguindo-se uma explicação sobre a arquitectura da aplicação e da datasets e para concluir como elas estão organizadas a estrutura da aplicação.

3.2 Conceitos

Nesta secção vão ser descritos conceitos úteis para a concretização do resultado da aplicação deste projecto.

3.2 Ferramenta de Programação

A criação de um modelo de aprendizagem de máquina envolve várias etapas e o uso de diferentes ferramentas dependendo do tipo de modelo, dos dados envolvidos e da preferência pessoal. Abaixo estão algumas das principais ferramentas comummente utilizadas no desenvolvimento de modelos de aprendizagem de máquina:

a. Linguagens de Programação: As linguagens de programação são essenciais para implementar algoritmos de aprendizagem de máquina e manipular dados. A linguagem para desenvolver modelo neste projecto foi Python: Extremamente popular devido à sua rica biblioteca de aprendizado de máquina, incluindo TensorFlow, Keras, PyTorch, Scikit-learn, entre outros.
b. Bibliotecas de Aprendizagem de Máquina: Existem várias bibliotecas em Python e outras linguagens de programação que fornecem implementações eficientes de algoritmos de aprendizado de máquina, tornando o desenvolvimento mais rápido e fácil. Algumas das bibliotecas mais conhecidas usadas no projecto:
    - TensorFlow e Keras: Frameworks populares para redes neurais e deep learning.

- PyTorch: Outro popular framework para deep learning e pesquisas em aprendizado de máquina.

c. Ferramentas de Visualização de Dados: A visualização dos dados é essencial para entender seu comportamento e distribuição. Segue algumas ferramentas de visualização usadas nos projectos:
   - Matplotlib: Uma biblioteca gráfica para criação de gráficos em Python.
   - Seaborn: Baseada em Matplotlib, fornece visualizações estatísticas mais atraentes e fáceis de usar.
   - Plotly: Uma biblioteca que permite criar visualizações interactivas e gráficos web.

d. Ferramentas de Pré-processamento de Dados: Antes de alimentar os dados ao modelo, muitas vezes é necessário pré-processá-los para lidar com valores ausentes, normalizar, padronizar, entre outras tarefas. Algumas ferramentas úteis usadas são:
   - Pandas: Biblioteca popular para manipulação e análise de dados em Python.
   - NumPy: Biblioteca para computação numérica em Python, frequentemente usada em conjunto com o Pandas.

e. IDE (Integrated Development Enviroment) ou Ambiente de Desenvolvimento: são ambientes interactivos que permitem combinar código, texto e visualizações em um único documento.. Das IDEs populares utilizadas no projecto são: Jupyter Notebook, Google Colab e Visual Studio Code. São úteis para explorar dados, criar protótipo modelos e compartilhar resultados.

3.2.1 Redes Neurais

São uma classe de modelos de aprendizagem de máquina que se inspiram na estrutura e funcionamento do cérebro humano. Os modelos de aprendizagem profunda, também conhecidos como redes neurais profundas, são um tipo especial de redes neurais que consistem em várias camadas ocultas, permitindo que o modelo aprenda representações complexas dos dados.

Existem vários tipos de redes neurais utilizadas para criar modelos de aprendizagem profunda, e vou descrever apenas as redes utilizadas no projecto:

- Redes Neurais Recorrentes (RNN): São projectadas para lidar com dados sequenciais, como texto, fala e séries temporais. As RNNs têm loops que permitem que as

informações sejam mantidas e propagadas ao longo do tempo, o que as torna adequadas para problemas de natureza sequencial.

- Long Short-Term Memory (LSTM): Uma variação de RNN, projetada para superar o problema do desvanecimento do gradiente (vanishing gradient) em sequências muito longas. As LSTMs são amplamente utilizadas para tarefas de processamento de linguagem natural (NLP) e outras tarefas sequenciais.

3.2.2 Dataset (conjunto de dados)

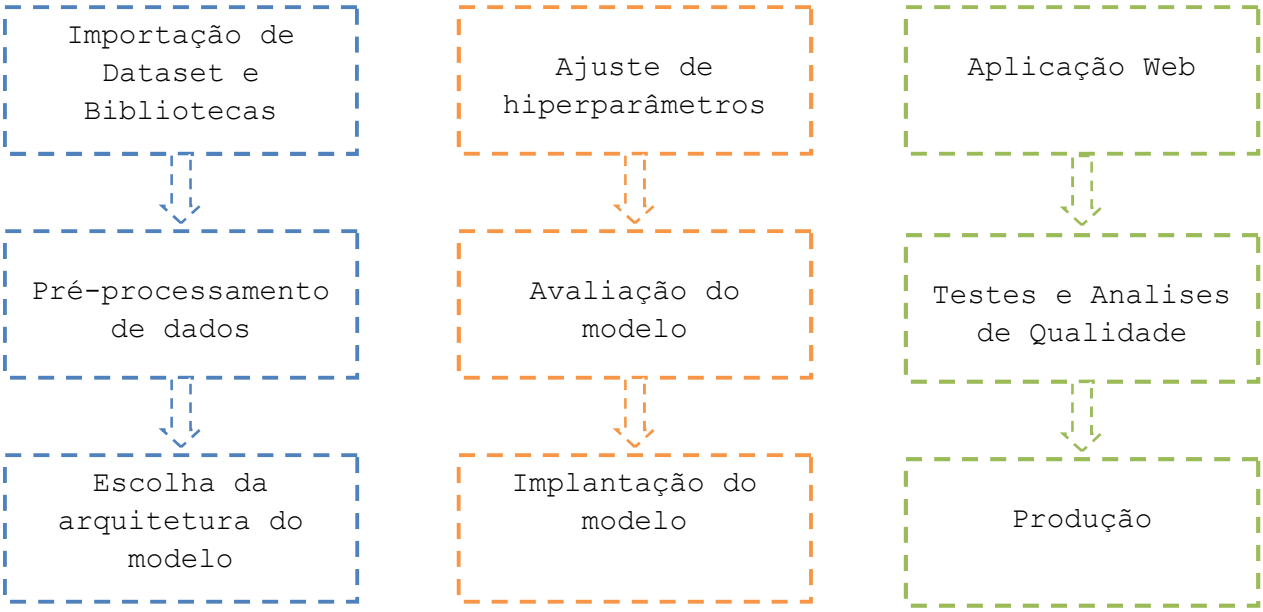
De extrema importância para criar um modelo de aprendizagem profunda de texto, ou qualquer modelo de machine learning ou deep learning em geral. O desempenho e a qualidade do modelo dependem, em grande parte, da qualidade, quantidade e representatividade dos dados utilizados no treinamento. Algumas razões pelas quais o dataset é crucial para criar um modelo de aprendizagem profunda de texto:

- Treinamento do Modelo: Os modelos de aprendizagem profunda requerem uma quantidade significativa de dados para aprender padrões e relações relevantes nos textos. Quanto mais dados tiver, mais bem informado será o modelo e poderá generalizar para novos exemplos.
- Representatividade: O dataset precisa ser representativo da tarefa que o modelo deve realizar. Se você estiver criando um modelo para classificar sentimentos em textos, por exemplo, é importante ter uma ampla variedade de textos que cubram diferentes tópicos, estilos de escrita e expressões de sentimento.
- Generalização: Um modelo bem treinado com um dataset representativo é mais propenso a generalizar para novos dados, ou seja, conseguirá fazer previsões precisas e úteis para textos que nunca foram vistos durante o treinamento.
- Redução de Viés: Um dataset bem construído também pode ajudar a reduzir viés indesejado no modelo. Isso significa que o modelo não se tornará parcial em relação a determinadas características dos textos ou grupos de dados específicos.
- Evitar Overfitting: Um dataset grande e variado pode ajudar a evitar o overfitting, que ocorre quando o modelo se ajusta muito bem aos dados de treinamento, mas não é capaz de generalizar para novos dados.
- Pré-processamento e Limpeza: O dataset é fundamental para o processo de pré-processamento e limpeza dos textos. Ao treinar um modelo de aprendizagem profunda de texto, é comum realizar tarefas como tokenização, remoção de stop words, lematização e outras etapas de preparação dos dados para melhorar a qualidade do treinamento.

- Avaliação do Desempenho: Após o treinamento do modelo, você precisa de um conjunto separado de dados (conjunto de teste) para avaliar o desempenho do modelo em exemplos não vistos. Um dataset adequado e bem dividido entre treino e teste é essencial para medir com precisão a performance do modelo.

3.3 Arquitectura do Sistema

Para criar um modelo de aprendizagem profunda de texto, você pode seguir uma arquitectura baseada em redes neurais. Existem várias abordagens e arquitecturas possíveis, mas aqui está um roteiro geral para criar um modelo de aprendizado profundo de texto:



3.3.1 Importação de dados

O primeiro passo é coletar uma vasta quantidade de texto de várias fontes, como livros, artigos, sites da internet e muito mais. Esses dados são usados para ensinar o modelo sobre a linguagem humana e como as palavras e frases são estruturadas.

3.3.2 Pré-processamento de Dados

Os dados importados passam por um processo de pré-processamento, onde são limpos, tokenizados e transformados em uma forma adequada para alimentar o modelo. Isso envolve dividir o texto em unidades menores, como palavras e sub-palavras, chamadas "tokens".

- Limpeza de texto: Remova pontuação, caracteres especiais, números e quaisquer símbolos indesejados.

- Tokenização: Divida o texto em palavras individuais (tokens) para criar sequências de tokens.
- Vetorização: Converta os tokens em representações numéricas, como vetores de palavras pré-treinados (por exemplo, Word2Vec, GloVe) ou embeddings treinados junto com o modelo.
- Sequência de padding: Pad as sequências para terem o mesmo tamanho, preenchendo com zeros ou cortando quando necessário.

3.3.3 Escolha da Arquitectura do Modelo

As redes neurais recorrentes (RNNs) são um tipo de arquitectura de rede neural projectada para lidar com sequências de dados, onde a ordem e a dependência temporal são importantes. Uma das variações mais populares de RNNs é a LSTM (Long Short-Term Memory), que é especialmente eficaz para lidar com problemas que envolvem sequências longas e dependências de longo prazo.

As LSTMs foram projectadas para superar uma limitação comum das RNNs tradicionais: o problema do desaparecimento do gradiente. Esse problema ocorre quando o gradiente (uma medida da inclinação da função de perda) diminui drasticamente à medida que é propagado de volta no tempo durante o treinamento da rede. Isso dificulta o aprendizado de dependências de longo prazo, já que a contribuição do gradiente diminui exponencialmente à medida que retrocedemos no tempo.

3.3.4 Treinamento do Modelo

Os dados pré-processados são usados para treinar o modelo. Isso envolve alimentar os dados pela rede neural, calcular as previsões do modelo e compará-las com os valores reais para calcular o erro. Algoritmos de otimização ajustam os parâmetros da rede para minimizar esse erro.

3.3.5 Ajuste de Hiperparâmetros

O processo de treinamento ocorre em várias iterações (épocas), ajustando gradualmente os parâmetros da rede e permitindo que o modelo aprenda representações mais ricas dos dados.

3.3.6 Avaliação do Modelo

Durante o treinamento, o modelo é testado em dados que ele não viu anteriormente para verificar seu desempenho. Métricas de avaliação são usadas para medir a qualidade das previsões do modelo.

3.3.7 Implantação do Modelo

Após um treinamento satisfatório, o modelo pode ser implantado para realizar tarefas específicas, como responder a perguntas, gerar texto, traduzir idiomas, entre outros. Integrado o modelo em sua aplicação Web para fazer previsões em novos dados de texto.

3.3.8 Testes e Análises de Qualidade

Realizar testes e análises de qualidade de um modelo em uma aplicação web é uma parte fundamental do desenvolvimento de software para garantir que o modelo esteja a funcionar correctamente, atendendo às expectativas e oferecendo uma experiência confiável aos utilizadores.

3.3.9 Aplicação Web

Escolhida a linguagem de programação Python no seu framework Flask sendo framework com uma estrutura completa para criar a interface simples e interactivo da aplicação web para utilizador.

3.3.10 Produção

Colocar em produção uma aplicação web para um modelo de classificação de documentos envolve várias etapas técnicas conforme foi mencionada acima.

CAPÍTULO 4

4.1 WEB INTERFACE

Neste capítulo será apresentada uma descrição o do processo de desenvolvimento do projecto, a começar com uma apresentação da tecnologia usada, seguindo-se uma breve explicação sobre a arquitectura da aplicação e da base de dados e para concluir como estão organizada a estrutura da aplicação.

4.1.2 Bootstrap

O Bootstrap é uma framework open-source utilizado no front-end para ajudar a desenvolver aplicações Web responsivas usando o metodo Responsive Web Design (RWD1). Para além de conter os vários elementos de Cascading Style Sheets (CSS), o Bootstrap apresenta uma diversidade de componentes em JavaScript e JQuery) que ajudam a implementar uma grande variedade funcionalidades, como por exemplo: tooltip2, menu-dropdown, modal, carousel, slideshow, etc. Esta framework é muito popular entre os designers, existindo, assim, varias bibliotecas e plug-ins para esta.

Na figura abaixo mostra o desenvolvimento da página principal do projecto com a estilização de Bootstrap para experiência do utilizador.

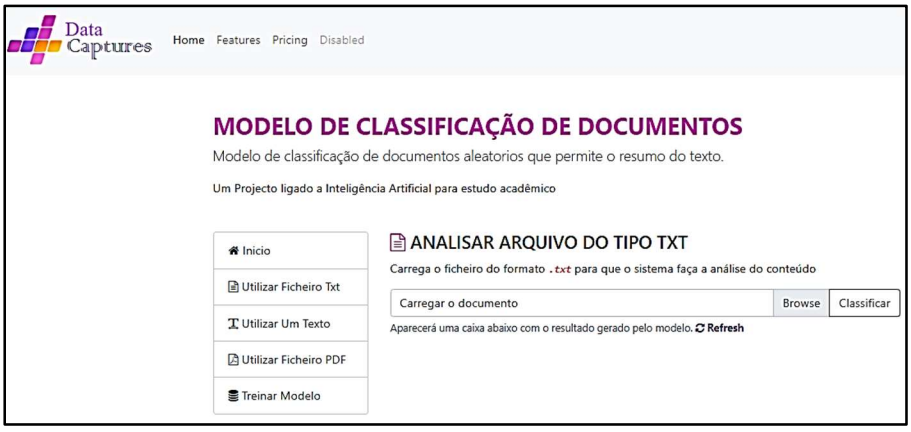


Figura 2 - Página Inicial

A página principal do projecto, apresenta três formas de classificação de documentos, isto é, valores de entrada para o sistema, que podem ser documentos no formato txt, documentos no formato pdf ou simplesmente textos escritos. O sistema consegue reconhecer o tipo de entrada de documento e fazer a sua análise e dar um resultado adequado ao utilizador. A figura a seguir mostra o resultado após a análise do documento, sendo como valor de entrada no formato de um txt.

Actualmente o sistema apenas suporta a língua inglesa, prevemos que nos próximos tempos voltaremos a dar suporte necessário para actualizar o sistema com mais línguas e com mais funcionalidades.

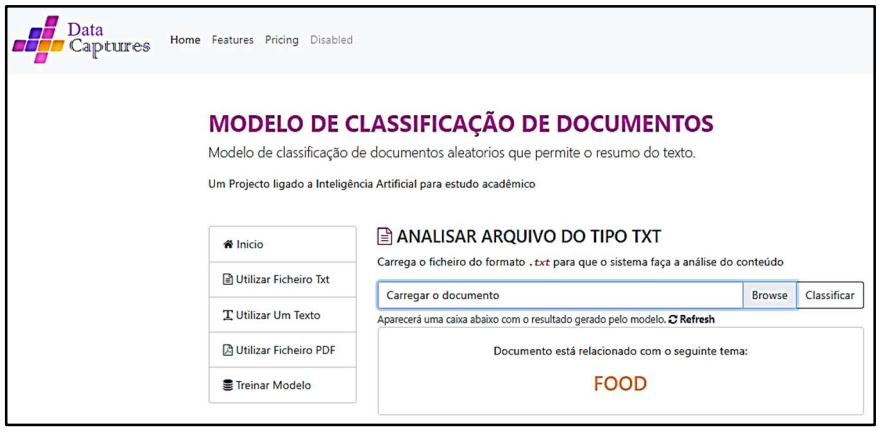


Figura 3 - Página Resultado

4.1.2 Framework Flask

O Flask é um framework leve e flexível em Python para criação de aplicativos web. Ele fornece as ferramentas necessárias para criar aplicativos web de forma rápida e eficiente, mantendo uma estrutura simples e flexível. Foi utilizado neste projecto para construir o backend.

4.1.3 Estrutura do Projecto

O Flask mantém uma estrutura fácil de entender no projecto. Normalmente, terá um directório principal para o *MMA Project* e, dentro dele, pastas para arquivos estáticos, templates, modelo, datasets e o próprio código do Flask.

```
MMA Project /
├── dataset/
├── model/
│   ├── documents/
│   └── create_model_from_trained.pkl
├── static/
│   ├── assets/
│   │   ├── dist/
│   │   │   ├── css/
│   │   │   │   └── bootstrap.css
│   │   │   ├── js/
│   │   │   │   └── bootstrap.js
│   │   │   └── style.css
│   │   └── font-awesome-4.7.0
│   ├── favicon.svg
│   └── logo.svg
├── templates/
│   ├── file/
│   │   └── index.html
│   ├── pdf/
│   │   └── index.html
│   └── text/
│       └── index.html
└── app.py
```

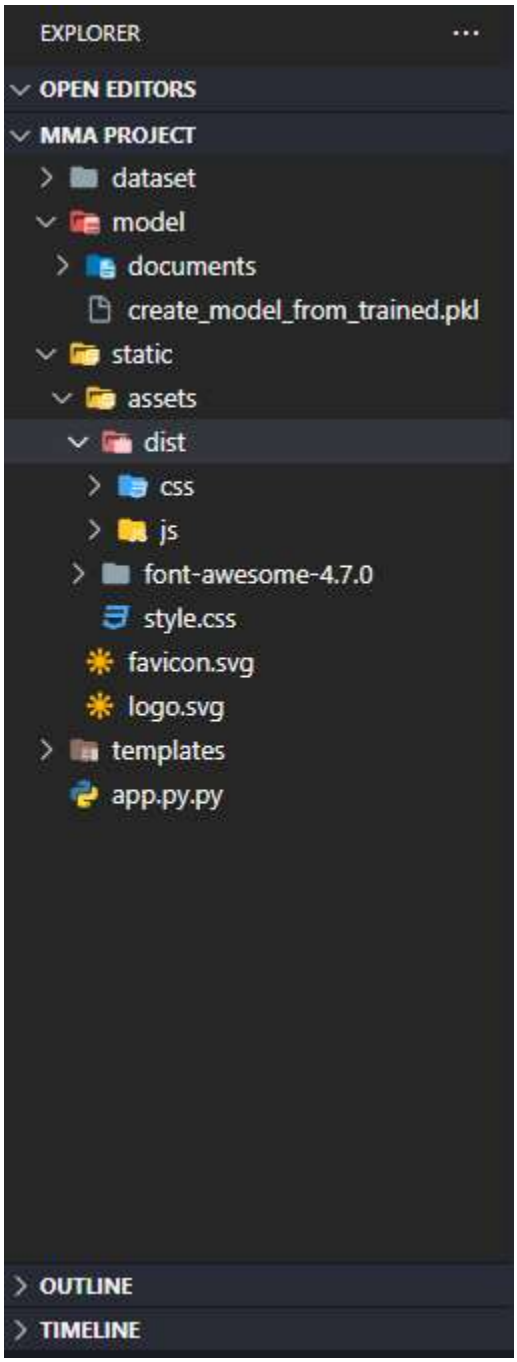


Figura 4 - Extrutura do Projecto

4.1.4 Importação de bibliotecas no Flask

Importar bibliotecas correctamente é uma parte fundamental ao trabalhar com modelos de aprendizagem profunda no Flask ou qualquer outra aplicação Python. Para utilizar o nosso modelo de aprendizagem profunda com o Flask, foi necessário importar algumas bibliotecas listados abaixo:

Vamos destacar apenas algumas das mais importantes bibliotecas utilizadas no projecto.

```
from flask import Flask, render_template, request
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
import re, os
from nltk.corpus import stopwords
```

Figura 5 - Importação de Bibliotecas

From flask é utilizada para importação da classe Flask que é uma estrutura de desenvolvimento web em Python.

Numpy é uma biblioteca fundamental para computação científica em Python. Ela fornece suporte para arrays multidimensionais (chamados de ndarray) e funções matemáticas de alto desempenho para manipulação desses arrays. NumPy é amplamente utilizado em análises de dados, visualização, processamento de sinais, álgebra linear, cálculos matriciais e em muitas outras áreas da computação científica e engenharia.

Pandas é uma biblioteca de software de código aberto amplamente utilizada para manipulação e análise de dados. Ela fornece estruturas de dados e ferramentas para lidar com dados tabulares, como tabelas e planilhas.

O ***TensorFlow*** é uma biblioteca de código aberto desenvolvida pela Google, amplamente utilizada para tarefas de aprendizado de máquina e redes neurais. Ele é especialmente popular para a criação, treinamento e implantação de modelos de aprendizado profundo, como redes neurais convolucionais e redes neurais recorrentes.

A biblioteca ***matplotlib*** permite que você crie uma variedade de gráficos, como gráficos de linhas, gráficos de dispersão, gráficos de barras, histogramas, gráficos de pizza e muito mais.

Keras é uma biblioteca de código aberto em Python, amplamente utilizada para o desenvolvimento de redes neurais e modelos de aprendizado profundo. Ela fornece uma interface de alto nível para construir, treinar e avaliar modelos de aprendizado profundo de maneira eficiente. Dentro dela podemos encontrar o ***keras.preprocessing.text*** é utilizada para pré-processar e transformar texto em sequências numéricas***,*** ***keras.model*** é utilizada para criar modelos de redes neurais de forma sequencial, camada por camada, ***keras.layers*** é um módulo da biblioteca Keras, que é uma API de alto nível para a criação e treinamento de redes neurais.

Import re (Regular Expressions) é utilizado para trabalhar com expressões regulares em Python. Expressões regulares são padrões de busca e manipulação de strings.

Import os (Operating System Interface), este módulo fornece uma interface para interagir com o sistema operacional subjacente. Ele permite que o acesso informações sobre o ambiente do sistema, manipule directórios, arquivos e caminhos de arquivo, execute comandos do sistema operacional e muito mais.

O módulo ***nltk***.corpus é parte da biblioteca Natural Language Toolkit (NLTK), que é amplamente utilizada para processamento de linguagem natural em Python. O submódulo ***stopwords*** do NLTK é utilizado para lidar com palavras de parada, também conhecidas como "stop words" em inglês.

5.5 Processamento de texto

Uma das coisas que garante o bom resultado do projecto de classificação de documentos é fazer com que o texto seja bem processado e com a mais alta qualidade. O modelo dependerá do texto para um resultado bom. Antes de começar a análise, é necessário pré-processar o texto. Isso envolve remover pontuações, normalizar o texto para lowercase, eliminar stopwords (palavras comuns sem significado), realizar stemming (reduzir palavras à sua forma raiz) e lematização (reduzir palavras à sua forma base). A figura abaixo mostra um trexo de código onde realizamos a remoção de pontuações e etc, do texto.

```
def clean_str(string):
    string = re.sub(r"[^A-Za-z0-9(),!?\'\`]", " ", string)
    string = re.sub(r"\'s", " \'s", string)
    string = re.sub(r"\'ve", " \'ve", string)
    string = re.sub(r"n\'t", " n\'t", string)
    string = re.sub(r"\'re", " \'re", string)
    string = re.sub(r"\'d", " \'d", string)
    string = ' '.join(word for word in string.split() if word not in
STOPWORDS)
```

Figura 6 - Função Clean, para Pre-Processamento do Texto

Posteriormente o desafio era transferir o texto em representações numéricas que os algorítmos possam entender. Isso é geralmente feito por meio de técnicas como TF-IDF (Term Frequency-Inverse Document Frequency) ou embeddings de palavras pré-treinados (como Word2Vec, GloVe, BERT) para o nosso caso utilizamos a biblioteca ***keras.preprocessing.text*** que facilitou atendendo a linhagem de bibliotecas que utilizamos conforme podem observar na lista acima.

Nesta mesma linhagem os dados foram dividas em conjuntos de treinamento e teste. Alimentei os dados de treinamento no algoritmo escolhido e ajustei os parâmetros do modelo para que ele possa aprender os padrões, relacionamentos entre os documentos e suas classes. As classes listadas no projecto são cerca de 18 classes conforme a figura abaixo:

```
labels = ['biology', 'geography', 'physic', 'accounts', 'computer',
'historical', 'math', 'science', 'politics', 'entertainment', 'crime',
'medical', 'graphics', 'business', 'technologies', 'space', 'sport',
'food']
```

Figura 7 - Categoria ou Classes Inicial

A figura a seguir mostra o código na função que cria um modelo de rede neural sequencial com uma camada de incorporação, uma camada de abandono espacial, uma camada LSTM e uma camada densa de saída. O modelo é compilado com uma função de perda apropriada e otimizador para tarefas de classificação de texto.

Definição da função model():

- Cria um modelo sequencial usando a classe Sequential do Keras.
- Adiciona uma camada de incorporação (Embedding) para converter sequências de palavras em vetores densos de números reais.
    - MAX_NB_WORDS: Número máximo de palavras únicas no vocabulário.

    - EMBEDDING_DIM: Dimensão dos vetores de incorporação.
    - input_length: Comprimento máximo da sequência de entrada (número de palavras em cada sequência).
- Adiciona uma camada de abandono espacial (SpatialDropout1D) para reduzir o overfitting.
    - Taxa de dropout: 20%.
- Adiciona uma camada LSTM (LSTM) com 128 unidades e dropout para regularização.
    - Número de unidades LSTM: 128.
    - Taxa de dropout para as conexões de entrada e saída: 20%.
    - Taxa de dropout recorrente: 20%.
- Adiciona uma camada densa (Dense) de saída com activação softmax.
    - NUM_CLUSTER: Número de clusters (classes) para classificação.
- Criação do modelo: *model = model()*
- Compilação do modelo:
    - Define a função de perda como categorical_crossentropy, que é a perda adequada para problemas de classificação com múltiplas classes.
    - Usa o otimizador Adam para ajustar os pesos da rede durante o treinamento.
    - Define métricas a serem monitoradas durante o treinamento, neste caso, a precisão (accuracy).

```
def model():
    model = Sequential()
    model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM,
input_length=X.shape[1]))
    model.add(SpatialDropout1D(0.2))
    model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(NUM_CLUSTER, activation='softmax'))

    return model

model = model()
model.compile(loss = 'categorical_crossentropy', optimizer='adam',
metrics = ['accuracy'])
print(model.summary())
```

Figura 8 - Função Leitura do Modelo

Foram definidas como paramêtro seguintes no para execução do modelo:

```
# The maximum number of words to be used. (most frequent)
MAX_NB_WORDS = 50000

# Max number of words in each complaint.
MAX_SEQUENCE_LENGTH = 250

NUM_CLUSTER= 18

EMBEDDING_DIM = 970
```

Figura 9 - Definição de Parâmetros para o Modelo

Resultado da execução do programa.

```
 0   Title     13226 non-null  object
 1   Document  13226 non-null  object
 2   Class     13226 non-null  object
dtypes: object(3)
memory usage: 310.1+ KB
Found 109904 unique tokens.
Shape of data tensor: (13226, 250)
Shape of label tensor: (13226, 22)
(11903, 250) (11903, 22)
(1323, 250) (1323, 22)
```

Figura 10 - Resumo dos Tokens Recolhidos

Na figura acima mostra a execução do código que processa um conjunto de dados de texto ou valores, identifica um grande número de tokens únicos, converte esses dados em tensores e lidando com um problema de aprendizado de máquina ou mineração de dados que envolve amostras com 250 atributos e 22 possíveis classes de etiquetas. Vou dividir a descrição em partes para facilitar a compreensão:

1. ***Memory Usage (Uso de Memória)***: Indica que o código está usando cerca de 310.1+ KB de memória. Isso pode ser relevante para monitorar o consumo de recursos durante a execução do código.
2. ***Found 109904 Unique Tokens (Encontrados 109904 Tokens Únicos)***: o código processou um conjunto de texto ou dados e identificou um total de 109904 tokens únicos. Isso inclui à contagem de palavras, termos ou símbolos distintos no conjunto de dados.
3. ***Shape of Data Tensor (Forma do Tensor de Dados)***: O conjunto de dados convertido em um tensor (uma estrutura de dados multidimensional frequentemente usada em computação numérica). A forma desse tensor é (13226, 250), o que sugere que

existem 13226 amostras (ou instâncias) no conjunto de dados, e cada amostra possui 250 dimensões. Isso poderia ser interpretado como uma matriz onde cada linha representa uma amostra e cada coluna representa um atributo.

4. ***Shape of Label Tensor (Forma do Tensor de Etiquetas)***: Similar ao tensor de dados, o conjunto de classes (ou rótulos) também foi convertido em um tensor. A forma desse tensor é (13226, 22), indicando que existem 13226 amostras e cada amostra possui 22 dimensões de classes.

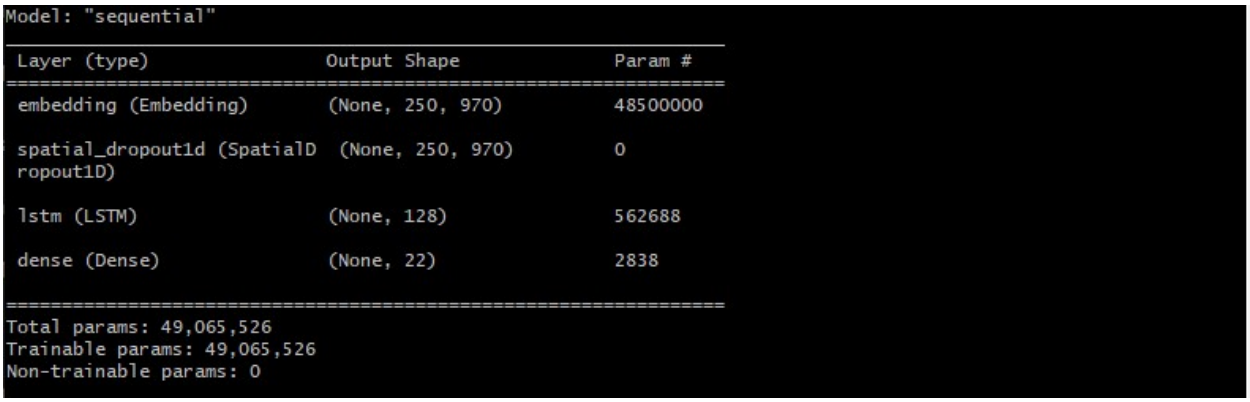


Figura 11 - Resumo Após Treinamento do Modelo

Na figura acima o modelo é configurado para processar sequências de entrada e produzir saídas de classificação em um problema específico. O uso de uma camada de Embedding ajuda a capturar o significado das palavras, a camada LSTM processa a sequência e a camada Dense gera as predições finais. O spatial_dropout1d auxilia na regularização para melhorar a generalização do modelo.

1. Camada de Embedding (Embedding):
    - Tipo: Embedding Layer
    - Entrada: Sequências de tamanho fixo de comprimento 250 (cada sequência representa uma série de palavras ou tokens).
    - Saída: Sequências de vetores de números reais de tamanho 970 para cada palavra/token na entrada.
    - Número de Parâmetros: 48.500.000
    - Função: Essa camada mapeia cada palavra/token em um espaço de vetor denso de dimensão 970, onde as relações semânticas entre as palavras podem ser

representadas de forma mais eficaz. Isso ajuda a capturar o significado e contexto das palavras nas sequências.

2. Camada de Spatial Dropout1D (SpatialDroppout1D)
    - Tipo: Spatial Dropout1D Layer
    - Entrada: Sequências de vetores de dimensão 970.
    - Saída: Sequências de vetores de dimensão 970.
    - Número de Parâmetros: 0
    - Função: Essa camada aplica um tipo de regularização chamado "dropout" às entradas. O dropout ajuda a evitar o overfitting, desativando aleatoriamente neurônios durante o treinamento para reduzir a dependência entre eles.
3. Camada LSTM (LSTM):
    - Tipo: Long Short-Term Memory (LSTM) Layer
    - Entrada: Sequências de vetores de dimensão 970.
    - Saída: Vetores de tamanho 128.
    - Número de Parâmetros: 562.688
    - Função: A camada LSTM é uma rede neural recorrente (RNN) especializada em processar sequências. Ela captura padrões de dependência ao longo do tempo em sequências de entrada. Nesse caso, a camada LSTM produz vetores de contexto (representações) das sequências de entrada.
4. Camada Dense (Dense):
    - Tipo: Fully Connected (Dense) Layer
    - Entrada: Vetores de tamanho 128 da camada LSTM.
    - Saída: Vetores de tamanho 22 (correspondendo às classes ou categorias de saída).
    - Número de Parâmetros: 2.838
    - Função: A camada densa é responsável por realizar a classificação ou predição final com base nas representações aprendidas nas camadas anteriores. Ela produz a saída final do modelo.

Total de Parâmetros: 49.065.526

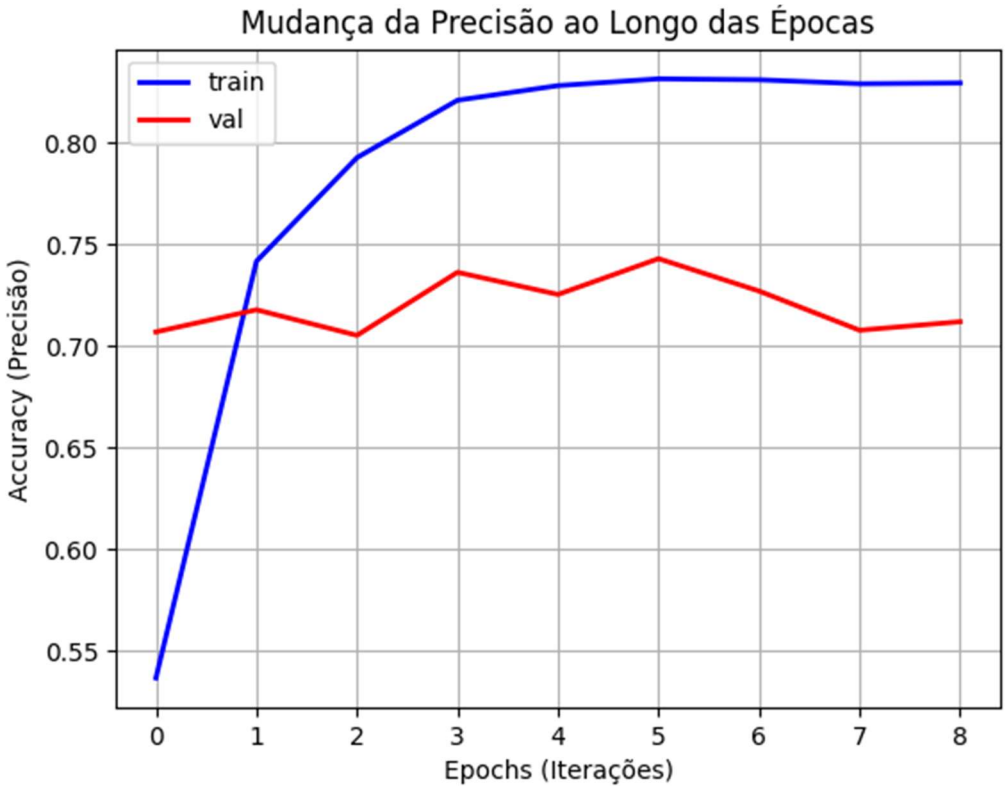


Figura 12 - Gráfico de Resultado da Precisão

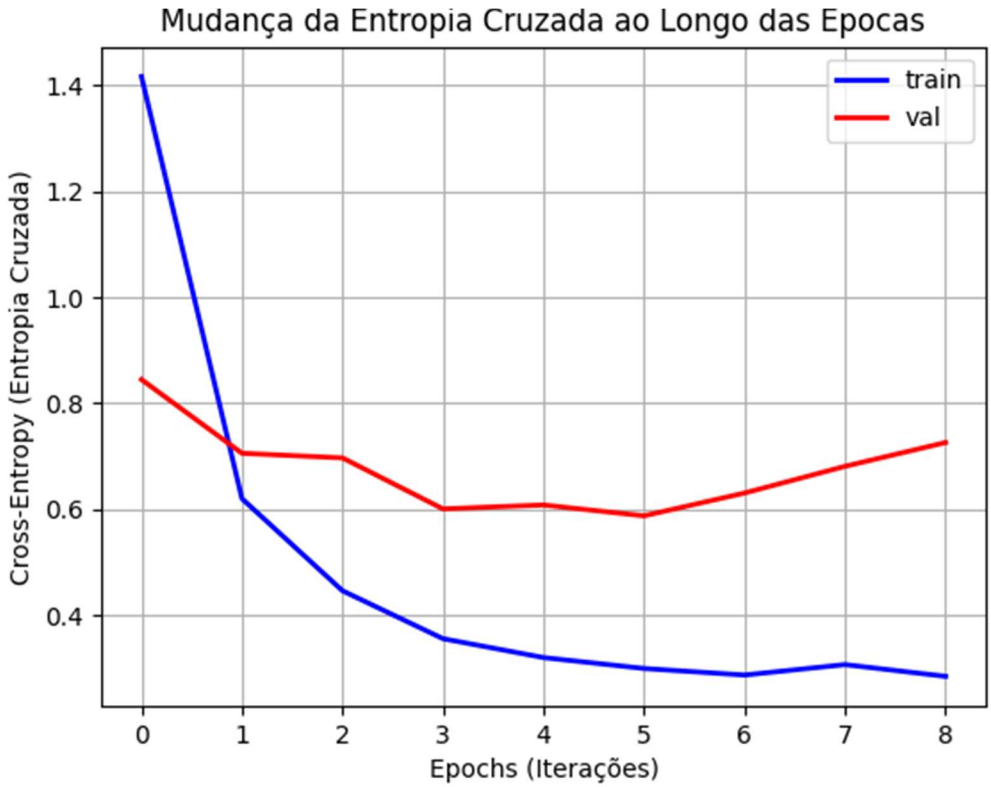


Figura 13 - Gráfico de Resultado da Entropia Cruzada

CAPÍTULO 5

5.1 CONCLUSÃO

Analisando os objectivos propostos inicialmente no capítulo 1.2, foi proposto a realização de uma aplicação Web que permita classificar documentos. Com esse objetivo em mente fez-se o levantamento de requisitos que a aplicação deveria ter e que foram descritos nos objectivos no capítulo 1.2. Detalhou-se igualmente os casos de uso descrevendo as funcionalidades que a aplicação deveria ter na secção 2.

Para implementação das funcionalidades da aplicação, criou-se um user case (casos de usos) através de ajuda de aplicação web *visual paradigma* que mostra a execução do modelo de classificação de documentos. Com ajuda de várias tecnologias que actualmente são usadas criou-se um aplicação verdadeiramente intuitiva e de fácil entendimento conforme foi mencionado na secção 3.2 – ferramentas utilizadas. Foi também adicionado um campo que possibilita o utilizador fazer a sua própria experiência criando assim novos modelos de treino.

Após a implementação da aplicação, efetuou-se testes de usabilidade com mais de 20 novos documentos para serem classificados. Os resultados destes testes foram positivo, tendo em media, sido concluídas com sucesso 89% das tarefas propostas. No entanto, os 11% restantes apresentaram resultados não esperados, nomeadamente a falta de compreensão do sistema, por ser unicamente na língua inglesa e texto não bem processado por alguma falha do sistema.

Relativamente a trabalho a realizar no futuro deverão ser considerados melhorias nos seguintes pontos:

- Incluir com mais detalhe o resultado, fazendo assim um resumo detalhado do codigo e não apenas título e novas línguas para classificação de documentos;
- Incluir com mais informações na barra inicial, como por exemplo: descrição de como utilizar a aplicação, termos e condições, contactos, etc;
- Concluir a opção de treinamento de novos modelos;
- Incluir filtros para caso de ajustar o resultado;
- Criação de outros modelos de classificação, seja ela imagens, fotos, identificação de documento de identidade, etc.

REFERÊNCIAS

[1] Aplicações na Web para validação e classificação de notícias pela crowd

[2] Solving Modern Document Classification Challenges With Deep Learning | SOTA Document Classification Use Cases

[3] Using pre-trained word embeddings - (https://keras.io/examples/nlp/pretrained_word_embeddings/)

[4] Flask's documentation – (https://flask.palletsprojects.com/en/3.0.x/)

[5] Document classification using deep neural network with different word embedding techniques – (https://www.inderscienceonline.com/doi/abs/10.1504/IJWET.2022.125654?journalCode=ijwet)

[6] Classification of Documents Using Convolutional Neural Network(CNN) – (https://medium.com/swlh/classification-of-documents-using-convolutional-neural-network-cnn-e0768bb81aad)

[7] - Redes Neurais Recorrentes — LSTM – (https://medium.com/@web2ajax/redes-neurais-recorrentes-lstm-b90b720dc3f6)

[8] - DIFERENÇA ENTRE ANN, CNN E RNN Rede Neural Artificial (ANN) – (https://acervolima.com/diferenca-entre-ann-cnn-e-rnn/)

[9] - REDES NEURAIS DEEP LEARNING COM TENSORFLOW - Revista Eletrônica Científica de Ciência da Computação (https://revistas.unifenas.br/index.php/RE3C/article/view/232)