Lucas Canossa Cipolla
Luís Carlos Gregório Pires

Analisador de Veracidade de Frases utilizando Processamento de Linguagem Natural

São Paulo, SP

2023

Lucas Canossa Cipolla
Luís Carlos Gregório Pires

Analisador de Veracidade de Frases utilizando Processamento de Linguagem Natural

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Profa. Dra. Anarosa Alves Franco Brandão

Coorientador: Paulo Pirozelli

São Paulo, SP

2023

Este trabalho é dedicado a todos aqueles que acreditam na importância da informação confiável e na luta contra a desinformação.
Às pessoas que estão em busca da verdade, que valorizam informações confiáveis e se comprometem em compartilhar e educar outras pessoas sobre esse tema crucial.
Por último, aos cientistas, que dedicam seus dias para avançar o conhecimento e permitir que a sociedade evolua com base em evidências sólidas.

Resumo

Este trabalho de conclusão de curso, desenvolvido por Lucas Canossa Cipolla e Luís Carlos Gregório Pires, apresenta o desenvolvimento de um analisador de veracidade de frases utilizando técnicas avançadas de Processamento de Linguagem Natural (PLN), incluindo o uso de modelos como BERT e Sentence-BERT, adaptando um modelo de Inferência de Linguagem Natural. O objetivo principal é combater a desinformação, diferenciando frases verdadeiras de falsas. O sistema desenvolvido avalia frases inseridas pelos usuários contra um banco de dados de frases verdadeiras, fornecendo um índice de veracidade e explicação contextual.

Os resultados dos testes, conforme indicado na Figura 11 do trabalho, mostram uma alta eficiência do modelo. Em um dos testes mais coerentes, o sistema obteve uma acurácia de 88,9% e uma sensibilidade de 99,3% quando desconsiderando informações não encontradas.

Palavras-chave: Processamento de Linguagem Natural, PLN, Transformers, Verificação de Informação, Análise de Veracidade, Sentence-BERT, BERT, Inferência de Linguagem Natural, ILN

Abstract

This graduation project, developed by Lucas Canossa Cipolla and Luís Carlos Gregório Pires, introduces the creation of a sentence veracity analyzer using advanced Natural Language Processing (NLP) techniques, including the implementation of models such as BERT and Sentence-BERT, adapting a Natural Language Inference model. The primary goal is to combat misinformation by distinguishing between true and false sentences. The developed system evaluates user-entered sentences against a database of true sentences, providing a veracity index and contextual explanation.

The test results, as indicated in Figure 11 of the study, demonstrate the high efficiency of the model. In one of the most coherent tests, the system achieved an accuracy of 88.9% and a recall of 99.3% when disregarding unfound information.

Keywords: Natural Language Processing, NLP, Transformers, Information Verification, Veracity Analysis, Sentence-BERT, BERT, Natural Language Inference, NLI

Lista de ilustrações

Listings

Sumário

1 Introdução

A proliferação de Fake News, ou notícias falsas, tem um impacto significativo na formação de bolhas de opiniões e na polarização da sociedade (VOSOUGHI; ROY; ARAL, 2018). Além disso, tais conteúdos prejudicam a construção de um conhecimento sólido sobre qualquer assunto, comprometendo o aprendizado e o acúmulo de informações confiáveis. A formação de bolhas de opiniões está intimamente ligada à polarização, uma vez que as pessoas são expostas principalmente a conteúdos que corroboram suas ideias, muitas vezes falsos. Essa dinâmica cria narrativas que alimentam a hostilidade entre diferentes grupos sociais, políticos e culturais, dificultando o diálogo construtivo e a busca por soluções comuns para os problemas sociais.

Essa polarização resultante das Fake News não apenas prejudica a saúde da democracia, mas também mina a confiança no processo de pesquisa científica (VICARIO et al., 2019). Além de limitar a pluralidade de opiniões, a propagação de informações infundadas enfraquece o processo científico, levando as pessoas a desacreditarem nas informações provenientes de pesquisas e artigos. A desinformação é tão disseminada que trechos de texto sem fontes confiáveis são mais facilmente aceitos do que pesquisas realizadas de forma rigorosa.

Essa situação demonstra a importância de combater a desinformação e promover a valorização do conhecimento embasado em evidências. Agências de verificação desempenham um papel fundamental nesse processo, ao analisarem o contexto das informações e determinarem sua veracidade. Esses meios de combate à desinformação têm como objetivo proteger o público em geral, fornecendo acesso a informações confiáveis e verificadas, além de desafiar a disseminação de conteúdos falsos. Somente com a promoção de uma cultura informacional responsável, baseada em fatos e evidências, é possível mitigar os efeitos negativos das notícias falsas na formação de opiniões polarizadas e na construção de um conhecimento sólido e confiável.

Além disso, as notícias falsas podem prejudicar os esforços de grupos de pesquisa no acúmulo de informações sobre um determinado tema. Um exemplo disso é o projeto KEML, cujo objetivo é desenvolver uma estrutura para agentes de conversação capazes de responder a consultas de alto nível ao longo do tempo sobre a Amazônia Azul, incluindo questões, argumentos, causas, explicações, inferências e planos relacionados a tarefas específicas. A ideia por trás do projeto é desenvolver tanto um agente conversacional responsável por esse tema, quanto técnicas gerais que não estejam vinculadas a um domínio específico. Um dos projetos desenvolvidos por eles foi o Pirá, um conjunto de dados bilíngue Português-Inglês para perguntas e respostas sobre o oceano (PASCHOAL et al., 2021).

1.1 Motivação

A motivação desse trabalho se encontra na tentativa de criar novas possibilidades na detecção de informações falsas na internet. Nos últimos anos, houve avanços significativos no campo do processamento de linguagem natural. Conceitos inovadores, como Transformers (VASWANI et al., 2017), modelos como BERT (DEVLIN et al., 2018) e outros Large Language Models (LLMs), têm possibilitado a automação da análise da linguagem natural, o que permite a execução automatizada de novas tarefas, como responder a perguntas (QA), solucionar problemas e, como neste caso, detectar conteúdos falsos.

Esses avanços tecnológicos têm fornecido ferramentas poderosas para lidar com a detecção de informações falsas, contribuindo para a promoção de um ambiente mais confiável e seguro no compartilhamento de informações. Ao utilizar essas técnicas avançadas de processamento de linguagem natural, espera-se que seja possível identificar e filtrar conteúdos fraudulentos, facilitando assim a disseminação de informações precisas e autênticas para o público em geral.

A aplicação dessas tecnologias no campo da verificação de informações pode trazer uma contribuição significativa para a sociedade, permitindo que os indivíduos tenham acesso a informações confiáveis e embasadas, além de promover a conscientização sobre a importância de verificar e questionar as informações encontradas na era digital. Com o avanço contínuo do processamento de linguagem natural, espera-se que a detecção de conteúdos falsos seja cada vez mais eficaz, auxiliando na proteção contra a disseminação de informações enganosas e no fortalecimento da confiabilidade na era da informação.

Como mostrado na pesquisa feita por Kumar e Shah (2018), modelos recentes de detecção de noticias falsas baseados em inteligência artificial possuem duas abordagens principais:

- Feature Engineering Based
- Propagation Based

Modelos baseados em Feature Engineering normalmente tentam capturar características textuais das noticias (número de palavras, título incoerente, estrutura da frase), ou características do usuário (tempo de registro, quantidade de publicações), conexão (referências corretas, numero de referências) e suporte (ocorrência de um artigo em outros artigos). Modelos baseados em propagação capturam a informações sobre o numero de compartilhamentos e encaminhamentos em meios de divulgação ou redes sociais.

Modelos baseados exclusivamente no conhecimento são escassos já que apresentam diversos desafios relacionados a manutenção da base de conhecimento com informações confiáveis e tratamento dos dados das noticias falsas já que estarão em diferentes formatos.

Um exemplo de modelo que tenta reconhecer noticias falsas a partir um banco com diversas Fake News (Fetaure Engineering Based) é o FakeBert (KALIYAR; GOSWAMI; NARANG, 2021), um modelo de detecção de noticias falsas baseado no modelo BERT (DEVLIN et al., 2018).

1.2 Objetivos

Dados os fatos e motivos, esse trabalho tem como objetivo construir um modelo de verificação de sentenças, baseado não na estrutura das noticias falsas, mas sim, em um banco de sentenças verídicas. Esse tipo de arquitetura possui duas vantagens:

- Possibilita a explicação do contexto do resultado
- Possibilita o fornecimento da fonte de onde a informação foi consultada.

A entrada do usuário será uma sentença, e a saída será um índice de veracidade, mostrando a probabilidade da veracidade daquela informação com base nos dados existentes na base construída. Além disso, se possível, será retornado a fonte e o trecho de texto de maior similaridade com o entrada do usuário. Para que o trabalho seja cumprido, precisamos passar pelas seguintes etapas:

- Construção de uma base estruturada de sentenças verdadeiras.
  - Extração de PDFs do Google Scholar, documentos de enciclopédias digitais e outras bases de sentenças sobre amazônia azul.
  - Preprocessamento das informações.
- Organização da arquitetura do sistema.
  - Projeto do Sistema.
  - Pesquisa de um modelo pré-treinado baseado no Sentence-Bert para calcular os embeddings de nosso banco de sentenças.
  - Pesquisa de um modelo pré-treinado de Natural Language Inference para cálculo de implicação e contradição entre sentenças.
- Teste e otimização do sistema.
- Criação de uma interface visual para o usuário.

Além disso, escolhemos o escopo da Amazônia Azul para reduzir a abrangência da construção do banco de dados. Essa escolha também visa possibilitar uma futura integração do projeto com o o agente conversacional desenvolvido pela equipe do KEML, um projeto desenvolvido pelo C4AI. (PIROZELLI et al., 2022)

1.3 Justificativa

A importância deste trabalho surge da necessidade de auxiliar no processo de verificação de informações, a fim de evitar a manipulação e os efeitos negativos que podem ser causados ao público em geral. O objetivo principal é assegurar a confiabilidade das informações presentes em um texto, utilizando avanços na área de processamento de linguagem natural para automatizar esse processo.

A tentativa de criar um sistema baseado em conhecimentos verdadeiras abre novas possibilidades de pesquisa, tentando solucionar desafios encontrados neste projeto. Isso inclui melhorias e formas de facilitar o processo de ajuste fino de modelos de PLN e o incentivo à criação de bases de dados de escopo específico para diferentes áreas.

O objetivo é criar um sistema completo e funcional, podendo entender os desafios encontrados na detecção de noticias falsas com base em conhecimento verdadeiro.

1.4 Organização do Trabalho

No Capítulo 2, serão apresentados os conceitos teóricos que embasam o desenvolvimento do projeto, juntamente com os modelos e trabalhos de literatura utilizados como referência. O objetivo deste capítulo é fornecer uma base teórica sólida para o trabalho. No Capítulo 3, será detalhado o método de desenvolvimento adotado para atingir os objetivos do projeto. Serão apresentadas as etapas necessárias para a realização do trabalho de forma eficiente e organizada. No Capítulo 4, serão descritos os requisitos necessários para o sucesso do projeto. Serão identificados as tecnologias, requisitos funcionais e não funcionais. No Capítulo 5, serão detalhados os passos cruciais no processo de implementação e execução do projeto. Explicaremos a abordagem específica escolhida e modelos utilizados para o desenvolvimento, destacando as razões por trás dessa escolha e como ela se alinha aos objetivos do projeto. Serão apresentados os desafios encontrados durante a implementação e as estratégias adotadas para superá-los. No Capítulo 6, serão apresentadas as conclusões finais, os resultados obtidos ao longo do trabalho e os próximos passos no avanço do tema. Serão discutidos os principais achados, as contribuições do projeto e as recomendações para futuras pesquisas e desenvolvimentos na área.

2 Aspectos Conceituais

2.1 Considerações Iniciais

Neste capítulo, serão apresentados os conceitos teóricos que fundamentam o projeto. Os conceitos e referências discutidos aqui são essenciais para uma compreensão completa do trabalho. Durante o desenvolvimento, foram pesquisados outros conceitos e referências, mas serão abordados apenas aqueles que estão diretamente relacionados ao projeto.

2.2 Amazônia Azul

A Amazônia Azul é uma região oceânica que compreende a superfície do mar, as águas sobrejacentes ao leito do mar, o solo e o subsolo marinhos contidos na extensão atlântica. A Amazônia Azul abrange a Zona Econômica Exclusiva (ZEE) e a Plataforma Continental (porção do fundo oceânico que margeia os continentes). A ZEE se estende por aproximadamente 200 milhas náuticas, ou seja, 370 km de extensão, ao longo de toda a costa brasileira, que possui um comprimento de 7.491 quilômetros. Nessa área, o Brasil possui direitos de soberania para a exploração, conservação e gestão dos recursos naturais presentes. (PEREIRA, 2019)

A área total da Amazônia Azul corresponde a aproximadamente 3,6 milhões de quilômetros quadrados, com possibilidade de extensão para até 5,7 milhões de quilômetros quadrados, o que é comparável, em ordem de grandeza, à superfície da Floresta Amazônica.

A Amazônia Azul é uma região de grande importância para o Brasil, pois abriga importantes recursos naturais, como petróleo, gás natural, peixes e biodiversidade. Além disso, a região é estratégica para o comércio exterior brasileiro, pois representa cerca de 95 por cento do tráfego marítimo nacional.

No contexto do presente trabalho, a Amazônia Azul é relevante pois é o tema sobre o qual o analisador de veracidade de sentenças irá avaliar a veracidade. As sentenças que serão avaliadas pelo analisador podem tratar de diversos aspectos da Amazônia Azul, como:

- *Recursos naturais: quantidade, localização, exploração e potencial de desenvolvimento.*
- *Biodiversidade: espécies, habitats, ameaças e conservação.*
- *Comércio exterior: volume, produtos, destinos e desafios.*
- *Estratégia militar: defesa, soberania e cooperação internacional.*

Escolhemos esse tema devido ao projeto KEML, que busca criar um agente conversacional de alto nível com conhecimento sobre Amazônia Azul, sendo capaz de responder perguntas com precisão e acurácia.

2.3 Processamento de Linguagem Natural

O processamento de linguagem natural (PLN) refere-se à área da inteligência artificial e da linguística computacional que se concentra na interação entre computadores e a linguagem humana. O surgimento da área remonta ao teste de Turing (TURING, 1950), que definia um critério de inteligência para máquinas, comparando-a com capacidades humanas. Após isso, um dos primeiros experimentos relacionados a Processamento de Linguagem Natural aconteceu em 1954, quando pesquisadores da universidade de Georgetown junto com a IBM conduziram um para a tradução automática de 60 frases do russo para o inglês. (JR., 1954)

O objetivo principal do PLN é permitir que os computadores compreendam, interpretem e gerem texto ou fala em linguagem natural de forma semelhante aos seres humanos. Os algoritmos utilizados no PLN têm a capacidade de aprender padrões e comportamentos presentes na comunicação humana, além de identificar informações relevantes ou conflitantes por meio da análise semântica, sintática e outras técnicas, como os modelos baseados em Transformers (VASWANI et al., 2017), como o GPT-3 (PENG et al., 2023) e o BERT (DEVLIN et al., 2018).

Desde o teste de Turing (TURING, 1950), houve questionamentos sobre a capacidade das máquinas de compreender e realizar atividades que os humanos fazem. Atualmente, algumas dessas atividades já podem ser executadas por máquinas, enquanto em outras, as máquinas podem auxiliar os humanos.

2.4 Embeddings

Embeddings, no contexto de processamento de linguagem natural, referem-se a representações numéricas de palavras em um espaço vetorial contínuo, figura 1. Essas representações são projetadas de maneira a capturar as relações semânticas e sintáticas entre palavras, permitindo que algoritmos de aprendizado de máquina entendam melhor a semelhança entre as palavras dada sua proximidade no espaço vetorial.

Um marco significativo na evolução dos embeddings é o modelo Word2Vec (MIKOLOV et al., 2013). A utilização de representações distribuídas de palavras em um espaço vetorial é benéfica para aprimorar o desempenho de algoritmos de aprendizado em tarefas de processamento de linguagem natural, ao agrupar palavras com características semelhantes.

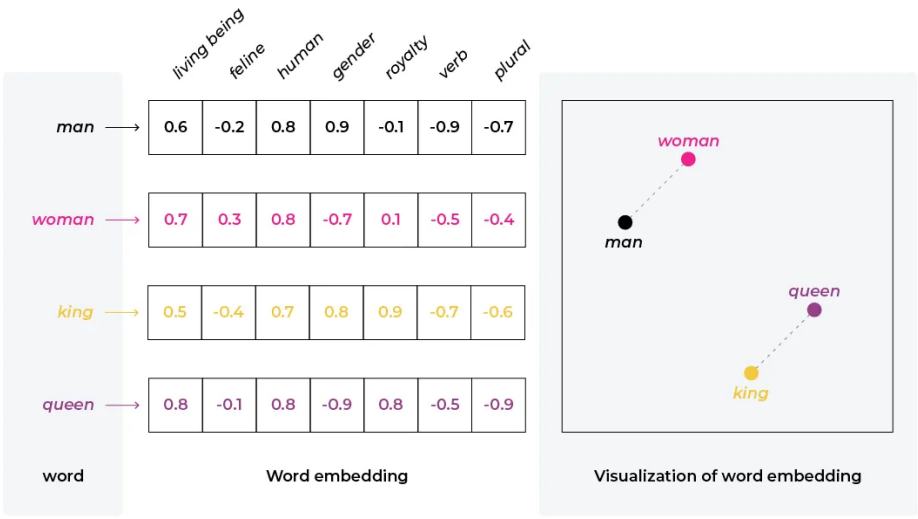

Figura 1 – Exemplo ilustrativo de Embeddings. (DHINAKARAN, 2022 (Acesso em 5 mar. 2023))

Outra evolução significativa foram modelos baseados em Transformers (VASWANI et al., 2017), como o BERT (DEVLIN et al., 2018) e, posteriormente, o Sentence-BERT (REIMERS; GUREVYCH, 2019). O Sentence-BERT permitiu que fossem calculadas representações vetoriais para sentenças em vez de palavras. Isso permite capturar semelhanças entre sentenças de uma forma mais eficiente e precisa.

2.5 Part-of-speech Tagging

O POS Tagging é uma técnica que consiste em atribuir classes gramaticais às palavras em uma sentença baseado na definição da palavra e sem seu contexto dentro da frase. Uma palavra pode representar diferentes classes gramaticais em sentenças diferentes. Tal tarefa tem diversas aplicações como: eliminação de ambiguidades, pré-processamento de texto, geração de texto, e análise de frequência de palavras e classes em um conjunto de textos.

2.6 Transformers

Os Transformers são uma arquitetura de rede neural que se destaca por sua eficácia no processamento de sequências de texto. Eles foram introduzidos em Vaswani et al. (2017) e desde então têm sido amplamente adotados em várias aplicações de PLN. Os Transformers podem ser resumidos como uma arquitetura do tipo Codificador-Decodificador baseada em Atenção.

O coração dos Transformers é o mecanismo de atenção, que permite que o modelo capture as relações entre as palavras em uma sequência. Ao contrário das arquiteturas

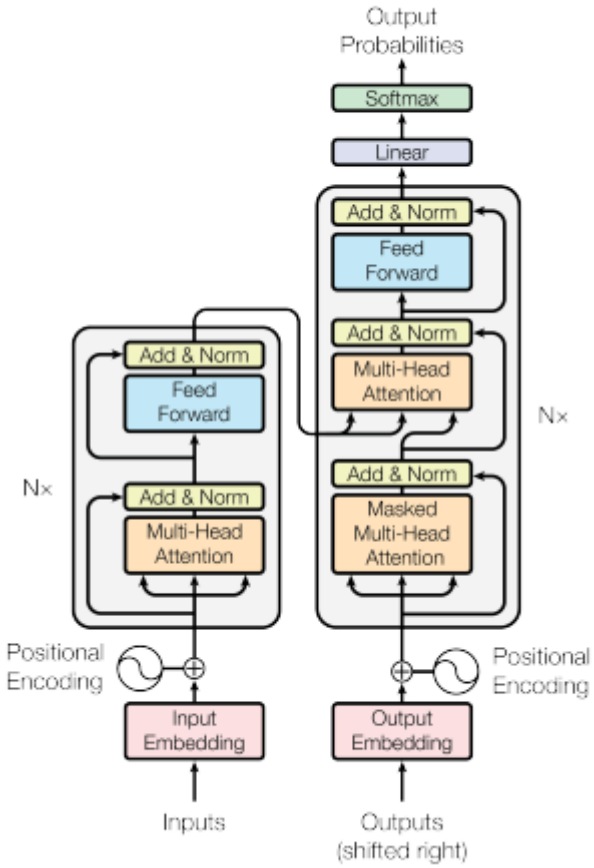

Figura 2 – Arquitetura do Transformers (Fonte: (VASWANI et al., 2017))

sequenciais tradicionais, os Transformers podem considerar todas as palavras de entrada simultaneamente, atribuindo pesos às palavras com base em sua importância relativa para a tarefa em questão.

A arquitetura original do Transformer, figura 2, é composta por um codificador e um decodificador. O codificador recebe a sequência de entrada e realiza a extração de recursos por meio de várias camadas de auto-atenção e redes neurais totalmente conectadas. O decodificador, por sua vez, gera a sequência de saída, usando atenção sobre o codificador e uma atenção auto-regressiva para garantir que cada palavra seja gerada corretamente, levando em consideração as palavras anteriores.

Uma característica crucial dos Transformers é a inclusão de informações de posição nas sequências de entrada. Para isso, são utilizados embeddings posicionais que codificam a ordem das palavras na sequência. Esses embeddings são adicionados aos embeddings de palavras para permitir que o modelo distinga as posições das palavras.

Uma outra característica interessante é a camada de Multi-Head Attention, figura 3. As matrizes resultantes de cada camada são projetadas linearmente em diferentes subespaços e aplicadas ao mecanismo de atenção paralelamente. Assim é possível capturar

diferentes relações entre as palavras.

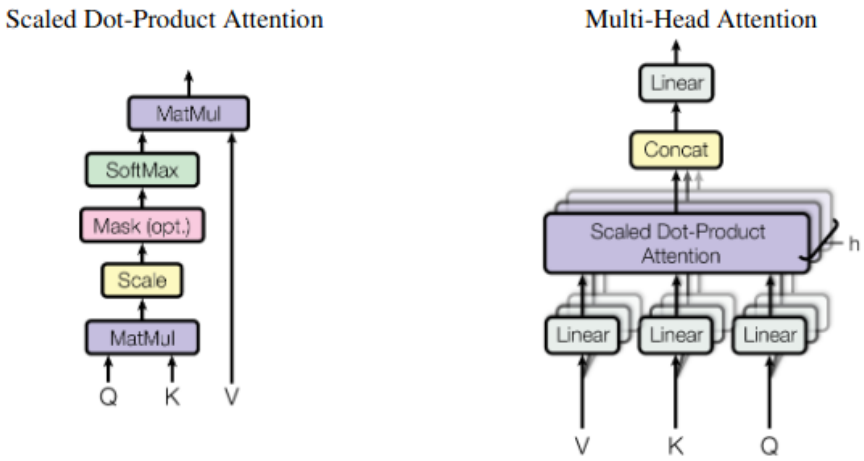

Figura 3 – Multi-Head Attention (Fonte: (DEVLIN et al., 2018))

Os Transformers (VASWANI et al., 2017) são compostos por várias camadas empilhadas. Cada camada é composta por um bloco de Transformers. A profundidade dessas camadas permite que o modelo capture informações complexas e de alto nível.

Os Transformers são treinados em grandes conjuntos de dados anotados, usando técnicas de aprendizado auto-supervisionado. Durante o treinamento, os parâmetros do modelo são ajustados para minimizar uma função de perda, que mede a discrepância entre as predições do modelo e os rótulos fornecidos. Após o treinamento, o modelo pode ser aprimorado em tarefas específicas com conjuntos de dados menores e rótulos específicos.

A arquitetura de Transformers tem sido aplicada com sucesso em uma ampla gama de tarefas de PLN, como tradução automática, sumarização de texto, resposta a perguntas, geração de texto e análise de sentimentos, entre outros. Sua eficácia e versatilidade tornaram os Transformers uma escolha popular para resolver problemas complexos de PLN.

2.7 BERT e Sentence-BERT

2.7.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) (DEVLIN et al., 2018) é um modelo de linguagem pré-treinado baseado na arquitetura Transformer, desenvolvido pelo Google. O BERT é amplamente utilizado em tarefas de Processamento de Linguagem Natural e tem sido um marco na área.

A arquitetura do BERT consiste em camadas empilhadas do codificador do Transformers. Isso permite que o treinamento seja verdadeiramente bidirecional, uma vez que,

no codificador, os tokens da sentença são processados simultaneamente em paralelo. O modelo é capaz de capturar o contexto das palavras considerando tanto o contexto à esquerda quanto à direita de cada token, o que melhora significativamente a compreensão das relações semânticas em uma frase. Além disso, com o Positional Embedding, é possível capturar a relação de ordem entre as sentenças.

O pré-treinamento do BERT é feito com duas tarefas não supervisionadas. A primeira é a **Masked Language Model (MLM)**. Tal tarefa consiste em mascarar uma porcentagem dos tokens de entrada, e prever qual seria esse token. A segunda é a **Next Sequence Prediction**, onde são inseridas 2 sentenças e o modelo precisa prever se uma sentença é a sequência da outra. Isso é útil em capturar a relação entre duas sentenças em tarefas como question answering.

Após o pré-treinamento, o BERT pode ser ajustado (fine-tuned) em tarefas específicas, como classificação de texto, reconhecimento de entidades nomeadas, resposta a perguntas, entre outras. Esse ajuste é realizado utilizando um conjunto menor de dados rotulados para a tarefa em questão. O modelo BERT pré-treinado é utilizado como base, e as camadas finais são treinadas com o objetivo de otimizar o desempenho na tarefa específica.

2.7.2 Sentence-BERT

Sentence-BERT (Sentence Embedding using BERT) (REIMERS; GUREVYCH, 2019) é uma abordagem específica que utiliza o modelo BERT para gerar representações semânticas de sentenças. Enquanto o BERT é originalmente projetado para trabalhar com pares de sentenças (como em tarefas de análise de similaridade ou paráfrase), o Sentence-BERT permite que o BERT seja usado para calcular representações de sentenças individuais.

Para encontrar o par de sentenças com maior similaridade utilizando apenas o BERT seria necessário passar todas as combinações de sentenças existentes ao modelo, o que levaria a uma sobrecarga computacional muito grande. O Sentence-BERT permite fazer isso de forma mais eficiente.

A ideia é passar apenas uma sentença como entrada do modelo BERT e agregar os vetores em um único vetor que represente o significado da sentença. Assim, para calcular a similaridade de duas sentenças, basta calcular a distância vetorial entre suas representações. Isso pode ser feito usando métricas de similaridade como a distância euclidiana, a distância do cosseno ou outras medidas de distância vetorial. Assim, em vez de passar todas as combinações, basta aplicar o modelo às sentenças e depois computar métricas básicas de distância.

O treinamento é feito usando duas redes neurais (BERT) siamesas, ou seja, mesmo

modelo e mesmos parâmetros, como é visto na figura 4. Ao fazer a inferência, basta calcular a representação vetorial da sentença de entrada.

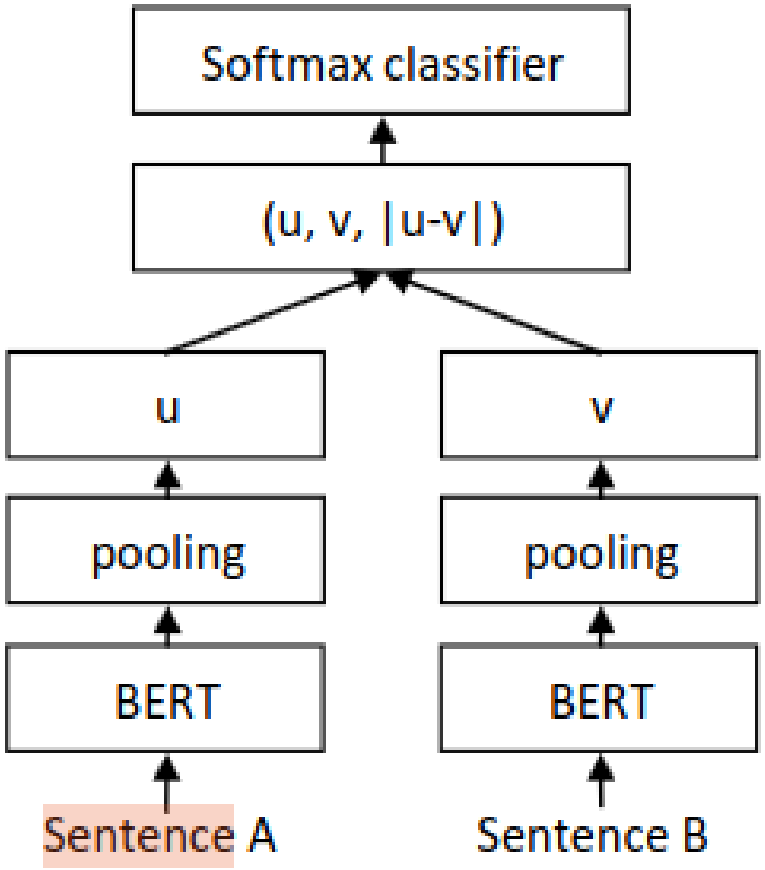

Figura 4 – Arquitetura do Sentence-BERT para classificação (Fonte: (REIMERS; GUREVYCH, 2019)

O Sentence-BERT pode ser utilizado para tarefas como cálculo de similaridade semântica, busca semântica em um conjunto de sentenças, Implicação textual (NLI), entre outras.

2.8 Natural Language Inference

Em processamento de linguagem natural, implicação textual, ou inferência de linguagem natural (ILN), é uma relação direta entre duas sentenças na qual uma das sentenças (premissa) acarreta a outra (Hipótese). O problema começou a ser estudado no campo de Processamento de linguagem natural a partir do Pascal RTE Challenge (DAGAN; GLICKMAN; MAGNINI, 2006), que surgiu em 2005 como uma forma de padronizar a

avaliação de modelos de NLI. Em resumo, o RTE Challenge envolve a análise de pares de textos para determinar se um deles pode ser inferido a partir do outro, com base na compreensão humana e no conhecimento de fundo comuns.

Atualmente, redes neurais que tentam resolver a tarefa de Inferência de Linguagem Natural geram uma distribuição de probabilidade para 3 labels (Entailment, Neutral, Contradiction). Esta tarefa pode ser feita utilizando Deep Transfer Learning a partir do modelo BERT. A técnica envolve a divisão do treinamento em duas fases (HOWARD; RUDER, 2018):

- *Pré-Treinamento*
- *Ajuste Fino*

O primeiro passo envolve o treinamento de um algoritmo em uma tarefa generalista para adquirir conhecimento de propósito geral e capturar nuances e padrões da linguagem. Essa é a função do BERT com o Masked Language Modeling (MLM) 2.7.1. A segunda etapa consiste em um treinamento subsequente para ajustar os pesos da rede a fim de lidar com tarefas mais específicas.

Normalmente, a última camada do BERT, responsável por prever a palavra na tarefa de MLM, é excluída e substituída por uma camada posterior para resolver um problema específico, como o de Inferência de Linguagem Natural.

Este foi o procedimento adotado em Laurer et al. (2022). Um ajuste fino foi realizado no modelo DeBERTaV3-base (HE; GAO; CHEN, 2023), uma versão aprimorada do modelo BERT. No ajuste, foram utilizados 1.279.665 pares rotulados em uma das três classes: entailment, contradiction ou neutral. O ajuste fino visa resolver a tarefa de Inferência de Linguagem Natural e os pares foram extraídos da base XNLI (CONNEAU et al., 2018), que contém pares premissa-hipótese em 15 línguas diferentes, incluindo o português.

Hoje existem diversos bancos de dados de benchmark para o teste de Natural Language Inference, um deles é o SNLI, uma coleção de 570k pares de sentenças em inglês escritas por humanos, manualmente categorizadas em implicação, contradição e neutras. (BOWMAN et al., 2015)

3 Método do Trabalho

Neste capítulo, serão definidas as fases principais do projeto, bem como estabelecidas as principais discussões acerca do desenvolvimento do sistema. Esta seção é importante para balizar o método e a sequência com que realizaremos as tarefas necessárias. Assim, poderemos, de forma organizada e objetiva, solucionar os problemas impostos nos objetivos.

3.1 Especificação de Requisitos e Pesquisa Bibliográfica

Durante a primeira etapa do projeto, conduziu-se uma pesquisa aprofundada sobre os avanços recentes na área de Processamento de Linguagem Natural (PLN). A investigação abrangeu publicações científicas, artigos de conferências, revistas especializadas e outros materiais relevantes disponíveis online.

O propósito dessa pesquisa foi compreender as últimas tendências, técnicas e abordagens no campo do PLN, especialmente no que tange à resolução de problemas relacionados à identificação de notícias falsas. Modelos de linguagem pré-treinados, como BERT, GPT-3, T5, entre outros, foram explorados, pois têm apresentado resultados promissores em diversas tarefas de PLN, como classificação de texto, sumarização automática, tradução automática, resposta a perguntas, inferência textual, entre outras.

A pesquisa das últimas tendências também abordou modelos e aplicações que poderiam auxiliar na construção de uma arquitetura capaz de identificar sentenças falsas com base em um banco de verdades conhecidas. Adicionalmente, foram levantados os requisitos necessários para a execução do modelo de PLN proposto, contemplando recursos computacionais, modelos relevantes e acesso a conjuntos de dados de alta qualidade para treinamento e validação do modelo. Aspectos não funcionais, como eficiência e confiança nas respostas do algoritmo, também foram considerados.

3.2 Coleta e Montagem do Banco de Dados

A coleta de dados e montagem do banco de dados é uma etapa crucial no projeto, pois é a partir dessas informações que o modelo de processamento de linguagem natural será treinado e validado.

Nesta parte do projeto, será realizada uma pesquisa mais específica para determinar quais serão as fontes utilizadas com os dados mais precisos e relevantes.

Além disso, na seção 5, discutiremos e descreveremos o processo e a implementação de coleta de cada fonte, além do processamento necessário para sua utilização. Serão

também discutidas e utilizadas técnicas para a coleta de diferentes tipos de fontes específicas, como PDFs, tabelas e textos.

3.2.1 Estrutura do Banco de Dados

Nesta parte, serão discutidas as possíveis estruturas de dados compatíveis com as tecnologias utilizadas no projeto. Isso será projetado junto com o resto do sistema como parte fundamental da estrutura.

Várias estruturas se mostram relevantes como um banco de sentenças verdadeiras. Bancos de documentos, bancos em modelo JSON ou organizados em tabelas, ou tabelas hash podem ser utilizados de forma eficiente.

3.3 Projeto do Sistema

O projeto do sistema envolverá a definição da arquitetura do software, escolha de tecnologias específicas, definição de interfaces de usuário e integração de módulos. Esta fase será orientada pelos requisitos identificados na etapa de especificação, visando criar um sistema robusto e eficiente para a identificação de notícias falsas.

Para a definição da arquitetura, será estudada uma abordagem que permita uma integração fluida dos componentes do sistema, garantindo eficácia e escalabilidade. O foco estará na seleção de técnicas de Processamento de Linguagem Natural (PLN) que possam ser aplicadas de maneira eficiente à identificação de informações falsas.

Na etapa de implementação, serão pesquisadas as melhores práticas para a construção eficiente dos módulos do sistema, com ênfase na aplicação prática das técnicas de PLN escolhidas. Além disso, serão analisadas bibliotecas que facilitem a implementação e integração entre os módulos, garantindo uma implementação coesa e eficaz.

Adicionalmente, serão definidos os modelos utilizados e as técnicas de processamento de linguagem natural mais relevantes para a resolução do projeto. Os modelos serão obtidos através da plataforma Hugging Face (4.3.4), proporcionando acesso a modelos pré-treinados e permitindo adaptações específicas para a detecção de notícias falsas.

3.4 Desenvolvimento da Interface

O desenvolvimento da interface será uma parte crucial do projeto, pois a usabilidade do sistema impactará diretamente na eficácia da identificação de notícias falsas. Serão estudadas as principais bibliotecas e aplicações disponíveis para interfaces visuais baseadas em dados, como Streamlit, Plotly e Matplotlib. A escolha será orientada pela necessidade

de apresentar as informações de forma clara e acessível, permitindo uma interação intuitiva por parte dos usuários.

Além disso, será dada atenção especial à implementação de recursos visuais que facilitem a compreensão das análises realizadas pelo sistema, contribuindo para a transparência e confiança na utilização da ferramenta. A interface será projetada de forma a simplificar a interação do usuário com as funcionalidades do sistema, garantindo uma experiência positiva durante o uso da aplicação.

3.5 Testes

Os testes visam verificar a eficácia e a performance do modelo em lidar com diferentes cenários, assegurando sua capacidade de fornecer resultados precisos e confiáveis.

Serão realizados testes de validação para verificar se o modelo está fornecendo resultados precisos e confiáveis, comparando os resultados obtidos pelo sistema em um conjunto de dados balanceado em sentenças falsas e verdadeiras. A geração desta base de sentenças será discutida, uma vez que não existem bases de dados atualmente com informações falsas sobre o escopo escolhido.

Durante os testes de validação, serão definidas as métricas mais relevantes para avaliar o desempenho do modelo no cumprimento do objetivo principal: identificar informações falsas. Estabelecer a importância de cada métrica é fundamental para o objetivo do modelo, permitindo a avaliação eficiente do projeto em relação ao propósito principal do sistema.

4 Especificação de Requisitos

Essa seção visa apresentar os requisitos necessários para o desenvolvimento do projeto, como tecnologias, bibliotecas, modelos, fontes de dados e requisitos funcionais.

4.1 Hardware

Para o desenvolvimento do algoritmo e realização de testes, foi utilizado um notebook Acer Nitro 5, modelo AN515-54. O sistema operacional empregado foi o Linux. O projeto foi desenvolvido em um notebook equipado com processador Intel i5 de 9ª geração e uma GPU NVIDIA GTX 1650 Ti.

A aplicação é executada em um sistema fornecido pelo Hugging Face, que possui 2 vCPUs e 16GB de memória.

4.2 Tecnologias

4.2.1 Python 3.11

O projeto será desenvolvido utilizando a linguagem Python, que é amplamente reconhecida e utilizada para projetos de inteligência artificial. Python oferece uma vasta gama de bibliotecas especializadas que podem auxiliar no desenvolvimento de modelos de IA, bem como na coleta e estruturação do banco de dados. A versão escolhida foi a 3.11, pois era a última versão com suporte lançada no início do desenvolvimento do trabalho.

4.2.1.1 Bibliotecas

As seguintes bibliotecas serão necessárias nas etapas de coleta de dados, pré-processamento dos dados, estruturação do banco de dados, modelagem dos modelos de inteligência artificial, entre outros.

requests: A biblioteca requests em Python é uma poderosa ferramenta para realizar requisições HTTP de forma simples e eficiente. Ela simplifica o processo de envio de solicitações e o recebimento de respostas HTTP, tornando mais acessível a interação com recursos web, como APIs e páginas da web. Tal biblioteca ajudará no processo de realizar o Web Scrapping das paginas da marinha e dos artigos do Google Scholar

BeautifulSoup: Beautiful Soup (RICHARDSON, 2007) é uma biblioteca Python usada para extrair dados de documentos HTML e XML. Ela fornece uma maneira conveniente de analisar e navegar pela estrutura desses documentos, permitindo que você

extraia informações específicas com facilidade. Beautiful Soup facilita a busca, filtragem e extração de dados de páginas da web, seja para realizar análises, coletar informações ou alimentar outras aplicações. A biblioteca suporta a análise de HTML malformado e se adapta a diferentes estilos de codificação. Ela também oferece recursos poderosos para pesquisa de elementos com base em critérios como tags, classes, ids, atributos e texto. Beautiful Soup é amplamente usada em web scraping e é uma escolha popular quando se trata de extrair dados de páginas web.

re: A biblioteca re (expressões regulares) em Python é uma biblioteca padrão que fornece suporte para trabalhar com expressões regulares. As expressões regulares são sequências de caracteres que definem padrões de busca em texto, permitindo realizar tarefas como correspondência, busca e substituição de padrões em strings. Essa biblioteca é útil para encontrar informações relevantes, ou fazer algum tipo de pré-processamento de dados.

lxml: A biblioteca lxml em Python é uma poderosa ferramenta para análise e manipulação de documentos XML e HTML. Ela é construída sobre a biblioteca C libxml2 e fornece uma interface Python eficiente para realizar operações como parsing a partir de uma string, navegação, busca e modificação em documentos XML e HTML.

wget: wget é uma ferramenta de linha de comando utilizada para baixar arquivos da web. O nome "wget"é derivado de "World Wide Web"e "Get". Essa ferramenta é amplamente utilizada em sistemas baseados em Unix e Linux, mas também está disponível para outros sistemas operacionais.

Pandas: Pandas é uma biblioteca que oferece estruturas de dados de alto desempenho e fáceis de usar, especialmente para análise de dados. Ela introduz duas estruturas principais: Series (uma matriz unidimensional rotulada) e DataFrame (uma estrutura de dados tabular bidimensional). Pandas permite ler, manipular, filtrar e visualizar dados de maneira conveniente.

transformers: A biblioteca transformers em Python é uma biblioteca de código aberto desenvolvida pela Hugging Face. Ela fornece uma coleção de modelos de linguagem pré-treinados e ferramentas relacionadas para tarefas de processamento de linguagem natural (NLP) usando arquiteturas de transformers, como o BERT.

TensorFlow e PyTorch: TensorFlow é uma biblioteca de aprendizado de máquina de código aberto desenvolvida pelo Google. A biblioteca PyTorch é uma biblioteca de aprendizado profundo de código aberto para Python, desenvolvida pela equipe de pesquisa do Facebook. Elas são utilizadas principalmente para criar e treinar modelos de aprendizado profundo, como redes neurais. Ambas as bibliotecas fornecem suporte nativo ao uso de CUDA para acelerar operações em GPUs da NVIDIA. Isso permite que você treine e execute modelos de forma eficiente em hardware compatível com CUDA.

Natural Langague Tollkit (NLTK) e Spacy: O Natural Language Toolkit, comumente conhecido como NLTK, é uma biblioteca popular em Python para processamento de linguagem natural (NLP). Ele oferece uma ampla gama de recursos e ferramentas para auxiliar no desenvolvimento de aplicações de NLP, tais como tokenização, stemming, lematização, marcação gramatical, análise sintática, classificação de texto, entre outros.

O Spacy também é uma biblioteca eficiente para o processamento de linguagem natural, auxiliando no desenvolvimento das mesmas tarefas. A biblioteca Spacy possui alguns modelos pré-treinados para a realização dessas tarefas.

Haystack: Haystack é um framework de código aberto fim a fim focado em desenvolver aplicações com Large Language Models (LLMs), ou outros modelos pré-treinados, baseados em sistemas de busca em uma grande coleção de documentos para diferentes casos de uso, como recuperação de informação, extração de texto e pesquisa de documentos. Haystack pode ser utilizado em aplicações de Question Answering (QA), Busca semântica ou Retrieval-Augmented Generation (RAG). Ele é construído de forma modular, possibilitando criar o processo desde a extração de dados, até a busca no banco de documentos. Além disso, por ser de código aberto, é possível modificar os códigos para atender especificidades do nosso projeto.

4.2.2 CUDA

CUDA (Compute Unified Device Architecture) é uma arquitetura de computação paralela desenvolvida pela NVIDIA. Ela permite que desenvolvedores utilizem as unidades de processamento gráfico (GPUs) da NVIDIA para realizar cálculos de propósito geral. Anteriormente, as GPUs eram usadas principalmente para renderização gráfica, mas com a introdução do CUDA, elas podem ser aproveitadas para realizar uma ampla gama de tarefas de computação de forma paralela, como simulações físicas, aprendizado de máquina, processamento de imagem.

4.2.3 Streamlit

Streamlit é uma biblioteca de código aberto em Python projetada para facilitar a criação de aplicativos da web interativos para análise de dados e visualização. Ela permite que os usuários criem rapidamente interfaces de usuário simples e eficientes, sem a necessidade de conhecimentos avançados em desenvolvimento web.

A principal ideia por trás do Streamlit é simplificar o processo de transformar scripts Python em aplicativos web interativos. Com apenas algumas linhas de código, é possível criar gráficos interativos, tabelas, widgets e outros elementos de interface de usuário, tornando o desenvolvimento de aplicativos acessível mesmo para aqueles que não têm experiência extensiva em desenvolvimento web.

4.3 Fontes de Dados

4.3.1 Google Scholar

O Google Scholar é um mecanismo de pesquisa de acesso livre para textos acadêmicos. Considera-se que o Google Scholar é o maior mecanismo de pesquisa desde 2018, estimando-se que contenha mais de 389 milhões de documentos.

Os pdfs foram coletados a partir de um algoritmo de Web scraping dos artigos, revistas e livros mais relevantes, possibilitando criar uma base de dados com base em textos científicos relevantes.

4.3.2 Marinha

O site da Marinha foi identificado como uma fonte relevante de informações e artigos relacionados à Amazônia Azul. Por meio de técnicas manuais, foram coletados dados textuais relevantes, como notícias, relatórios e documentos disponíveis nesse site. Essa fonte foi importante principalmente nas etapas iniciais, onde desenvolvemos uma versão inicial para testar a funcionalidade do sistema.

4.3.3 BLAB Wiki

A BLAB-wiki (PIROZELLI et al., 2022) é uma enciclopédia de conteúdo especifico desenvolvido pelo projeto KEML baseados em conhecimentos de pesquisadores da área e páginas verificadas de enciclopédias digitais como o Wikipedia.

Em Pirozelli et al. (2022), é destacado que a falta de textos abrangentes sobre tópicos básicos relacionados à Amazônia Azul foi um desafio no desenvolvimento dos módulos BLAB (sigla em inglês para Blue Amazon Brain). Tal questão também foi um desafio para esse projeto, já que não existem fontes estruturadas e confiáveis de informação pública sobre a Amazônia Azul.

Em resposta a essa lacuna de informações acessíveis, os autores decidiram criar sua própria pequena enciclopédia chamada BLAB-Wiki. Esta wiki é escrita por especialistas no campo e fornece informações básicas sobre a Amazônia Azul para leitores leigos, com base na literatura científica.

A BLAB-Wiki está organizada em quatro eixos principais: socioambiental, biodiversidade, físico-químico e legislação e governança. Cada eixo discute diferentes aspectos da Amazônia Azul, como poluição marinha, efeitos das mudanças climáticas nos oceanos e ecologia das profundezas marinhas.

Tal biblioteca é a principal fonte de informações de qualidade, sendo verificadas por especialistas na área. Além disso, é a fonte de informação que possui a maior quantidade e

diversidade de informações.

4.3.4 Hugging Face

O Hugging Face é uma plataforma e comunidade voltada para o desenvolvimento e compartilhamento de modelos de linguagem natural e recursos relacionados à inteligência artificial. Essa plataforma desempenha um papel significativo na disponibilização de fontes de dados e modelos pré-treinados para diversas tarefas de processamento de linguagem natural.

O Hugging Face é uma valiosa fonte de informação já que fornece banco de dados existentes sobre diversos tipos de informações. Neste projeto, ele foi fonte de pesquisa para dados relacionados à Amazônia Azul.

Ao longo de pesquisas sobre fontes estruturadas sobre Amazônia Azul, encontramos uma base a qual vinha sendo desenvolvida pelo próprio C4AI para o KEML[1]. Esta base contém aproximadamente 200 mil frases sobre a legislação da região. No decorrer do projeto, a base foi temporariamente retirada do ar, uma vez que está em processo de construção pelo grupo e ainda não foi completamente validada. No entanto, identificamos um número significativo de sentenças válidas nela.

Além das fontes mencionadas, outras fontes de dados relevantes podem ser exploradas durante a coleta, dependendo das necessidades e requisitos específicos do projeto. É importante ressaltar que todas as práticas de coleta de dados serão realizadas em conformidade com as políticas de privacidade, direitos autorais e outras diretrizes aplicáveis.

4.4 Requisitos Funcionais

Nesta seção, apresentaremos os requisitos funcionais do projeto, abrangendo casos de uso, interfaces e funcionalidades.

4.4.1 Funcionalidades

As duas principais funcionalidades incluem uma busca semântica na entrada do usuário em um banco de sentenças, visando encontrar as sentenças mais semelhantes, e um modelo capaz de determinar a veracidade de uma sentença com base em outra. Além disso, é essencial contar com uma interface que permita a interação do usuário com o sistema.

Juntamente com as funcionalidades do sistema, são necessários alguns passos prévios, como a montagem do banco de dados, envolvendo coleta, pré-processamento dos textos e estruturação do banco de dados, e passos posteriores, como testes e otimização.

[1] Mais informações em: <https://c4ai.inova.usp.br/pt/pesquisas/>

Além disso, o banco de dados, o algoritmo de verificação e a interface do usuário devem estar adequadamente interligados.

4.4.2 Interface do Usuário

Para viabilizar a interação dos usuários com o sistema, é fundamental desenvolver um software que permita a inserção de uma sentença a ser testada. Esse software deve fornecer o resultado do modelo, indicando se a sentença é verdadeira ou não, junto com informações contextuais e a fonte da comparação em relação à sentença inserida pelo usuário.

Adicionalmente, é necessário que o software seja rápido e eficiente para não impactar a experiência do usuário. Para alcançar esse objetivo, é preciso escolher uma aplicação que atenda às necessidades do sistema e estratégias de armazenamento em cache para partes que demandam maior capacidade de armazenamento, como o banco de dados e os modelos.

5 Desenvolvimento do Trabalho

5.1 Construção da Base de Sentenças Verdadeiras

5.1.1 Extração dos dados

Uma das primeiras dúvidas do projeto foi a definição de verdade e como poderíamos obter uma fonte confiável para a base de conhecimento. Essa base precisaria ser escalável, pois à medida que a informação sobre um escopo específico aumenta, precisaríamos incorporar as novas informações relevantes sobre o tema em nossa base de dados. Além disso, enfrentamos a dificuldade da disponibilidade de informações estruturadas existentes sobre a Amazônia Azul, as quais estão dispersas em artigos, livros e revistas em diferentes formatos.

Nossa primeira ideia foi utilizar sites de instituições oficiais brasileiras, como o da Marinha Brasileira ou do Ibama. Inicialmente, o site da Marinha brasileira mostrou-se mais promissor, apresentando uma variedade de artigos e publicações relacionados à Amazônia Azul, área pela qual é responsável pela proteção, manutenção e preservação.

A primeira tarefa executada foi a coleta manual de alguns artigos, revistas e periódicos da Marinha para validar a construção do banco de dados. Os textos estavam em formatos PDF, o que acrescentou um desafio ao projeto: a extração de dados de PDFs de maneira concisa e coerente, evitando perdas. O problema residia no fato de que cada texto seguia uma estrutura diferente. Para possibilitar uma escalabilidade futura na quantidade de textos, não poderíamos restringir a coleta apenas a uma estrutura específica. A abordagem inicial foi utilizar aplicações existentes de conversão de PDF para texto e a separação das sentenças por pontuação. Entretanto, tal abordagem trazia muitos erros. Ainda não existem, ou pelo menos não foram encontradas, aplicações boas e genéricas o suficiente de conversão de PDF, mesmo as baseadas em Reconhecimento ótico de Caracteres (OCR).

Entretanto, continuamos pesquisando outras fontes de dados e formas de conversão de texto, sem que a informação viesse tão suja. Assim, chegamos a mais três fontes de dados:

- Google Scholar
- BLAB-wiki
- Hugging Face

O Google Scholar, seção 4.3.1, é uma plataforma desenvolvida pela Google, onde se encontram informações sobre artigos científicos. Nela, você pode pesquisar um tema específico, e a plataforma retorna todos os artigos que envolvem aquele tema. Pesquisando sobre a Amazônia Azul, encontramos a maioria dos artigos relevantes da Marinha que havíamos encontrado anteriormente.

O Google Scholar se comporta como um site, e coletar os artigos científicos um a um não tornaria nosso sistema facilmente escalável. Assim, desenvolvemos um código que realiza a mineração dos primeiros 50 documentos mais relevantes sobre o tema no site, em formato PDF. Com isso, é possível obter um arranjo suficiente de conhecimento científico para nossa base de verdades. Neste caso, quanto mais informações pudermos ter, melhor para a arquitetura geral do projeto, porém seguimos um balanço ao qual não custasse tanto ao desempenho do sistema e não coletasse artigos irrelevantes, podendo fazer a verificação da sentença em até 1 minuto na maioria dos casos.

Nosso código faz a coleta da página do Google Scholar, buscando o link de download do PDF de todos os artigos da página. Isso é feito fornecendo quantas páginas do Google Scholar devemos buscar e a palavra chave à qual queremos pesquisar. Logo após, abrimos um arquivo em formato .txt para armazenar estes links ao qual faremos a coleta. Em seguida, fazemos a requisição de cada página do Google Scholar armazenando o seu HTML com o método .get() da biblioteca requests. Em seguida, o conteúdo referente ao arquivo de cada documento é buscado e inserido em nosso arquivo de links.

O passo seguinte é iterar sobre os links encontrados e fazer o download do conteúdo desse link no formato PDF com o método download da biblioteca wget. Assim, teremos um arranjo dos 50 PDFs mais relevantes sobre o tema que definirmos, no nosso caso Amazônia Azul. Essa é uma das principais forças para o projeto ser escalável, dado que podemos construir uma base com verdades científicas sobre qualquer tema, e atualizar de tempos em tempos as informações mais relevantes encontradas.

A segunda fonte de dados é a BLAB-wiki, conforme mencionado na seção 4.3.3. Esta fonte foi fornecida pelo projeto KEML. É importante destacar que a BLAB-wiki é considerada a fonte mais confiável que utilizaremos, uma vez que foi redigida e revisada por pesquisadores especializados em áreas relacionadas à Amazônia Azul. Os textos foram fornecidos em formato txt, e só foi necessário separá-los em sentenças.

A terceira e última fonte foi encontrada no site Hugging Face, na seção 4.3.4. Tal fonte é um conjunto de perguntas e respostas em formato de tabela para o treinamento de tarefas de question answering sobre a legislação da Amazônia Azul. Este conjunto é uma construção do próprio C4AI para o projeto KEML[1] e hoje não está mais listada publicamente pois está passando por uma curadoria, porém é um volume baixo de sentenças

[1] Disponível em: <https://c4ai.inova.usp.br/pt/pesquisas/>. Acesso em: 12 dez. 2023

as quais podem estar comprometidas, sendo ainda uma boa estrutura para nosso projeto. Ela foi baixada em formato .parquet e convertida em uma lista de sentenças. Utilizamos a coluna de resposta `new_long_answers`, já que parecia ser a coluna que fornecia a maior quantidade de informação.

Ao final da primeira etapa da extração, tínhamos alguns arquivos txt, referentes a BLAB-wiki, uma lista de sentenças com as informações da base de QA e 50 PDFs com artigos retirados do Google Scholar.

5.1.2 Haystack: DocumentStore e Conversão de arquivo

Durante este processo, nos deparamos com a biblioteca Haystack. Como mostrado na Seção 4.2.1.1, tal biblioteca possui módulos desde a extração de dados até a busca semântica e o cálculo da implicação. Utilizamos essa biblioteca em todo o desenvolvimento do nosso projeto, devido à facilidade de implementação e por ser de código aberto, possibilitando a flexibilização e mudanças de algumas classes e métodos.

É importante destacar que a biblioteca Haystack mantém seus dados organizados em bancos orientados a documentos , chamados `DocumentStore`. A `DocumentStore` do Haystack é projetada para facilitar a construção de sistemas de recuperação de informação em grandes conjuntos de documentos. Ela serve como um repositório para armazenar e recuperar documentos textuais, armazenados em um banco SQL.

Além disso, a biblioteca possui integração com o Facebook AI Similarity Search (JOHNSON; DOUZE; JÉGOU, 2019) (FAISS). O FAISS é uma biblioteca eficiente para busca por similaridade e clusterização em grandes conjuntos de dados. Isso é útil, pois podemos calcular a similaridade entre nossos embeddings de forma eficiente, trazendo escalabilidade para o aumento de nosso banco de dados.

Os documentos armazenados são estruturados em classes, como mostrado a seguir:

Listing 5.1 – Estrutura da classe Document

```
class Document:
content: Union[str, pd.DataFrame]
content_type: Literal["text", "table", "image"]
id: str
meta: Dict[str, Any]
score: Optional[float] = None
embedding: Optional[np.ndarray] = None
id_hash_keys: Optional[List[str]] = None
```

O campo `content` armazena o texto guardado, no nosso caso, as sentenças. `content_type` é o tipo de conteúdo armazenado, no nosso caso, `text`. `id` é o identificador do documento na `DocumentStore`. `meta` são metadados dos documentos, não utilizados, por enquanto. O campo `score` representa a similaridade da sentença após o cálculo com a sentença de entrada.

`embedding` representa a representação numérica da sentença em um espaço vetorial contínuo, utilizado para calcular a similaridade, por enquanto nulos. E, por fim, o `id_hash_keys` é um atributo utilizado para gerar um id único para o documento, se o id não for fornecido.

Para conversão e separação dos documentos em sentenças utilizamos as seguintes Classes:

- *FileConverter*
  - *PDFToTextConverter*
  - *TextConverter*
- *PreProcessor*

As classes `PDFToTextConverter` e `TextConverter` são herdeiras da `FileConverter`, que é encarregada da conversão de arquivos de variados formatos para objetos da classe `Document`. Enquanto a `PDFToTextConverter` lida com a conversão de PDFs para texto antes de armazená-los como objetos `Document`, a classe `TextConverter` faz a conversão do texto para o mesmo formato de objeto.

A conversão de PDF para texto é baseada na biblioteca PyMuPDF, uma biblioteca Python de alto desempenho para extração, análise, conversão e manipulação de dados de documentos PDF (e outros). Apesar da extração não ser perfeita, devido às diferentes estruturas de PDFs coletados, foi possível fazer uma limpeza posterior para selecionar apenas sentenças que consideramos válidas. Utilizamos essas classes para converter nossas fontes em uma lista de documentos através do método `convert()`.

Dessa forma, teremos três listas de documentos, ainda não separadas por sentença.

5.1.3 Pré-Processamento

Após decidirmos sobre a estrutura do banco de dados, extrairmos os dados das fontes escolhidas, precisávamos separá-las em sentenças e realizar uma limpeza das sentenças que não traziam nenhum significado, como títulos e sentenças mal convertidas.

O primeiro passo foi separar os documentos em documentos menores, com o conteúdo sendo apenas as sentenças. Isso foi feito através da classe `PreProcessor`.

Listing 5.2 – Exemplo de uso da classe PreProcessor

```
1 preprocessor = PreProcessor(
2     language="pt",
3     clean_empty_lines=True,
4     clean_whitespace=True,
5     clean_header_footer=False,
6     split_by="sentence",
```

```
7       split_length=1,
8       split_respect_sentence_boundary=False)
```

Tal classe é utilizada para realizar limpezas gerais nos textos, removendo espaços em branco e linhas vazias desnecessárias, além de ter a capacidade de eliminar cabeçalhos e rodapés repetitivos. Adicionalmente, essa classe é responsável por separar os documentos em sentenças. Para isso, ela utiliza um modelo de tokenização de palavras da biblioteca NLTK, conforme apresentado na Seção 4.2.1.1, pré-treinado para capturar os tokens que indicam os limites da sentença. Para pré-processar os textos, basta passar a lista de documentos ao método `process()` como argumento, e será retornada uma lista de novos documentos, com o conteúdo das sentenças.

A última etapa do pré-processamento consistiu em selecionar apenas as frases que consideramos válidas. Optamos por utilizar somente frases verbais, ou seja, aquelas que contêm verbos em sua estrutura. Isso evita a inclusão de títulos, subtítulos e frases mal convertidas a partir dos PDFs.

Essa etapa não foi realizada com o auxílio da biblioteca Haystack, mas sim por meio da técnica de POS-Tagging. Utilizou-se o modelo `pt_core_news_lg` do spacy, que possui 97% de acurácia em tarefas desse tipo[2]. Foi criado um código que adiciona a informação de se existe verbo na sentença como um metadado de cada documento. Enfim, foram filtradas apenas as sentenças que possuíam verbos, a fim de obter apenas sentenças carregadas de significado.

Por fim, foi criada a FaissDocumentStore, passando como o cálculo de similaridade será feito (`similarity`), a dimensão do modelo de embeddings que utilizaremos (`embedding_dim`), e o link para a conexão SQL onde a base será armazenada. Todos os documentos foram escritos na base através do método `write_documents()`.

Listing 5.3 – Instância da classe FAISSDocumentStore

```
1 document_store = FAISSDocumentStore(
2     similarity="cosine",
3     embedding_dim=512,
4     sql_url = "sqlite:///final_faiss_document_store.db")
```

No final, obtivemos uma base separada em sentenças em uma estrutura de dados organizada e eficiente. Um fluxograma do processo pode ser visto a seguir:

5.2 Projeto e Implementação

Com a base de sentenças estruturada, a segunda etapa do projeto consistiu em desenvolver uma arquitetura capaz de identificar sentenças falsas, comparando-as com

[2] Disponível em: <https://spacy.io/models/pt>. Acesso em 1 nov. 2023

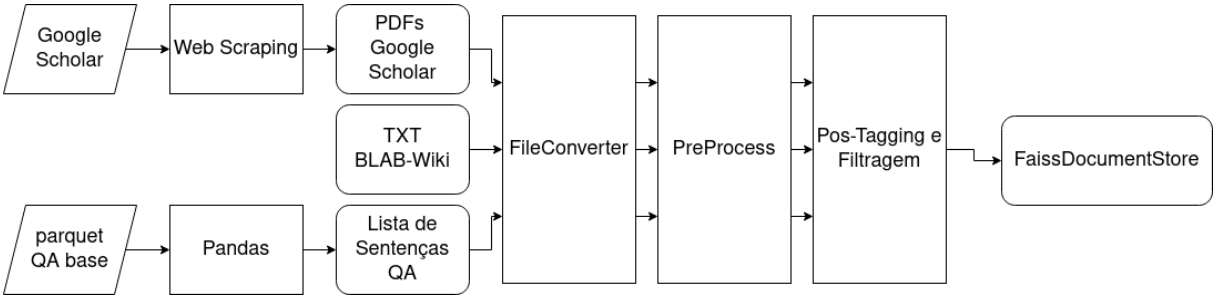

Figura 5 – Fluxograma da extração dos dados e criação da base de dados. (Fonte: Elaborado pelo Autor)

uma base de sentenças verdadeiras. A principal dificuldade foi conceber um método para atribuir uma relação de confirmação ou contradição ao comparar duas sentenças.

5.2.1 Arquitetura do Sistema

A solução encontrada envolve adaptar a tarefa clássica de inferência de linguagem natural para resolver o problema de veracidade de informações. Isso foi feito utilizando dois modelos pré-treinados de linguagem natural distintos. Um deles é responsável pela busca semântica, enquanto o outro é o modelo de inferência de linguagem natural, o qual atribui uma relação de implicação, contradição ou neutralidade entre duas sentenças. A hipótese subjacente é que se uma sentença for falsa, ela contradirá as sentenças presentes no banco de verdades. Caso a sentença seja verdadeira, será possível inferir, de alguma forma, essa sentença a partir das sentenças do banco.

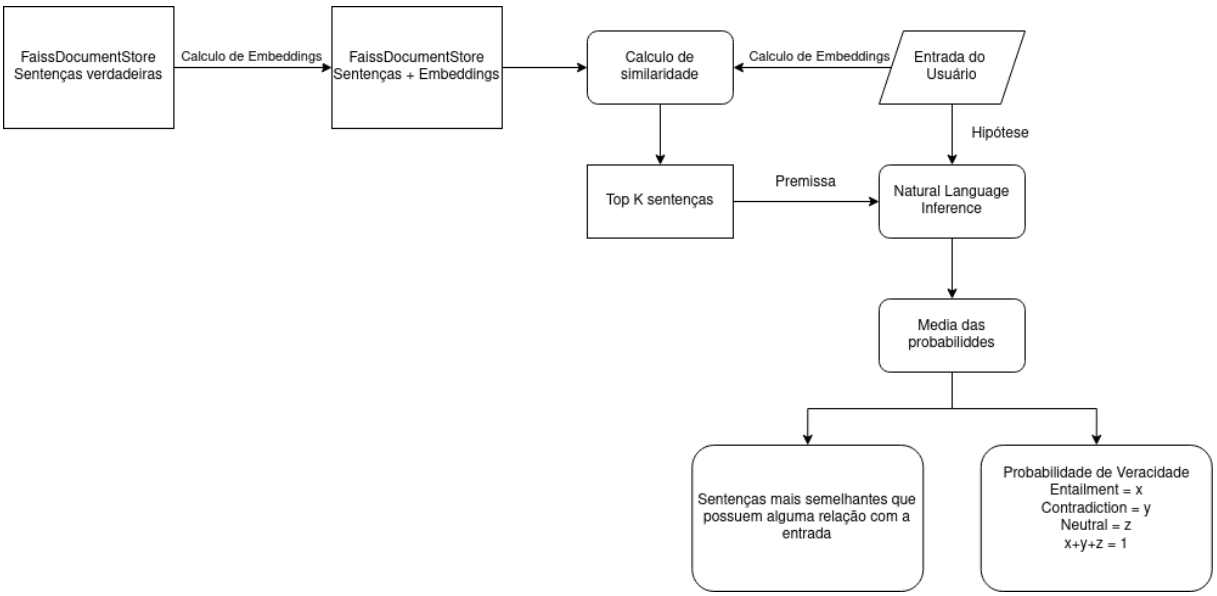

Figura 6 – Arquitetura da Aplicação. (Fonte: Elaborado pelo Autor)

Primeiramente, foi necessário escolher um modelo capaz de comparar duas sentenças semanticamente, seja diretamente, como no modelo BERT, Seção 2.7.1, seja através da

distância vetorial de seus embeddings, como apresentado na Seção 2.4 (Sentence-BERT, Seção 2.7.2). Dessa forma, seria possível realizar a busca semântica na base de sentenças. Após calcular as sentenças mais similares, efetuamos a inferência do modelo de Natural Language Inference, calculando a probabilidade de implicação, contradição ou neutralidade entre a entrada e as sentenças escolhidas na busca. O passo final consiste em calcular uma agregação entre os resultados obtidos para as diferentes sentenças e retornar um resultado final, juntamente com as sentenças que foram consultadas, proporcionando uma confiabilidade maior no resultado.

5.2.2 Cálculo de Embeddings

Ao iniciarmos a busca por um modelo de comparação semântica, nos deparamos inicialmente com o BERT (2.7.1). Para calcular a similaridade entre a entrada e todas as sentenças no banco, seria necessário submeter a entrada junto com cada sentença do banco ao modelo no momento em que o usuário inserisse a sentença. Entretanto, isso se mostra computacionalmente inviável.

Posteriormente, encontramos o Sentence-BERT (2.7.2), reconhecido como estado da arte para o cálculo de similaridade entre sentenças. A abordagem é simplificada: pré-calcular os embeddings do banco de sentenças. Assim, quando o usuário utiliza o sistema, é necessário calcular apenas o embedding da sentença de entrada e realizar o cálculo da distância por cosseno em relação às sentenças do banco. Essa abordagem é computacionalmente mais eficiente do que a inferência direta de um modelo.

Conforme discutido na Seção 2.7.2, o Sentence-BERT consegue capturar nuances da linguagem durante o pré-treinamento, graças ao mecanismo de atenção. No entanto, é importante notar que o pré-treinamento do Sentence-BERT é conduzido exclusivamente na língua inglesa.

Dessa forma, optamos por adotar o modelo multilíngue que compartilha a mesma arquitetura do Sentence-BERT, conforme apresentado em Yang et al. (2019). Esse modelo utiliza o SNLI (BOWMAN et al., 2015) — similar ao Sentence-BERT — porém é traduzido para outras 15 línguas, incluindo o português, através do Google Translator. A dimensão do embedding calculado pelo modelo é 512.

Dentro da arquitetura deste projeto, o cálculo é realizado em dois momentos distintos. No primeiro momento, procedemos à inserção de cada sentença de nossa base de sentenças verdadeiras de forma individual. Esse procedimento nos permite compilar um banco de dados contendo os embeddings de todas as frases verdadeiras.

No segundo momento, quando o usuário insere sua sentença, realizamos o cálculo do embedding exclusivo para essa sentença em particular. Posteriormente, este embedding é comparado com os embeddings presentes em nosso banco de sentenças verdadeiras,

utilizando a medida de similaridade do cosseno. Este processo visa determinar a similaridade semântica entre a sentença fornecida pelo usuário e as sentenças autenticadas em nosso banco de dados.

5.2.3 Natural Language Inference

Na nossa arquitetura, selecionamos o modelo multi lingual pré-treinado por Laurer et al. (2022) para resolver a tarefa de inferência de linguagem natural, como explicado em 2.8.

Este modelo é utilizado após a busca semântica, inferindo se a frase fornecida pelo usuário é implicativa, contraposta, neutra ou não contém informações em relação à base.

5.2.4 Implementação

Na Seção 5.1, explicamos como o Haystack, Seção 4.2.1.1, nos auxiliou a estruturar e implementar nossa base de dados como uma `FaissDocumentStore`, permitindo uma eficiente busca semântica.

Para realizar a busca semântica, precisamos primeiramente calcular os embeddings. Para isso, existe uma classe no Haystack chamada EmbeddingRetriever, responsável por fornecer o modelo que calcula os embeddings e realizar efetivamente a busca semântica.

Listing 5.4 – Instância da Classe EmbeddingRetriever e Cálculo dos Embeddings

```
1 retriever = EmbeddingRetriever(
2 document_store=document_store,
3 embedding_model="sentence-transformers/distiluse-base-multilingual-cased
    -v1")
4
5 document_store.update_embeddings(retriever)
```

Em sua instância, é necessário fornecer a `DocumentStore`, onde estão armazenadas as sentenças, e o modelo escolhido, como mostrado na Seção 5.2.3. É importante destacar que os modelos estão todos disponíveis no site Hugging Face, Seção 4.3.4, e disponível para download na biblioteca transformers, 4.2.1.1, a qual possui integração com o Haystack. O Haystack é responsável por buscar e baixar o modelo para o uso no sistema.

Calculamos os embeddings de todas as sentenças presentes na base com o método `update_embeddings()`, como mostrado em Listing 5.4. No fim, possuímos uma base de sentenças verdadeiras, com os embeddings calculados. Para que o sistema não precise sempre calcular os embeddings da base, salvamos a base através do método `save()` salvando em um documento `.faiss`. Assim, quando precisarmos utilizá-la, basta utilizar o método `load()`.

Quando formos carregar novamente nossa `FaissDocumentStore`, precisamos instanciar novamente o `EmbeddingRetriever`, pois ele será o responsável por efetuar a busca semântica.

A ultima classe que utilizaremos do Haystack será o `EntailmentChecker`, responsável por realizar a inferência de Linguagem Natural. 5.2.3. O Haystack possui já implementada tal classe, entretanto, decidimos modificá-la para que se adequasse melhor ao nosso projeto. Este é o beneficio de escolher uma biblioteca de código aberto. Além de entender a lógica por trás da função, é possível realizar modificações que se adéquem ao nosso projeto.

Listing 5.5 – Instância da classe EntailmentChecker

```
entailment_checker = EntailmentChecker(
    model_name_or_path="MoritzLaurer/mDeBERTa-v3-base-mnli-xnli",
    entailment_contradiction_consideration=0.7,
    entailment_contradiction_threshold=0.95,
    similarity_threshold=0.75,
)
```

Em resumo, realiza-se uma busca semântica nas `K` sentenças mais semelhantes à sentença submetida, utilizando o `EmbeddingRetriever`. Nos primeiros testes manuais, percebemos que, muitas vezes, as sentenças retornadas, considerando apenas a ordem decrescente de pontuação de similaridade, incluíam algumas que não eram tão semelhantes semanticamente. Desse modo, decidimos adicionar um limiar de similaridade. Assim, é necessário que a pontuação de similaridade gerada pelo Sentence-BERT seja maior que o limiar estabelecido em `similarity_threshold`. Então, o `EmbeddingRetriever` seleciona as `K` sentenças mais parecidas, e o `EntailmentChecker` faz um filtro da pontuação de similaridade.

Após a seleção das sentenças mais similares, a classe realiza a inferência do modelo de NLI. Para cada sentença retornada pelo `EmbeddingRetriever`, é criada uma distribuição de probabilidade para as labels entailment, contradiction e neutral. Aqui, fizemos mais uma alteração ao código original da classe `EntailmentChecker`. Como queríamos considerar apenas sentenças que possuíssem uma relação de implicação ou contradição relevante, decidimos criar um limiar de consideração em relação à probabilidade retornada para cada label. Tal campo é definido em `entailment_contradiction_consideration`. Ou seja, se a pontuação de contradição ou implicação for superior ao campo `entailment_contradiction_consideration`, a sentença é considerada. Caso contrário, a próxima sentença na lista é avaliada. Nesses casos, a sentença entra na lista que montará a pontuação final, onde terá peso igual às outras sentenças. Após a análise de cada sentença considerada, é calculada a média das probabilidades para cada categoria.

Adicionalmente, durante o processo de análise de cada sentença, se a probabilidade de implicação ou contradição agregada entre as sentenças for superior ao limiar especificado em `entailment_contradiction_threshold`, nenhuma outra sentença é considerada. Isso ocorre porque há fortes indícios de que a sentença submetida está relacionada de forma significativa,

seja por implicação ou contradição, com a base de dados. Essa funcionalidade já existia na classe original.

Os limiares, apesar de excluírem algumas sentenças e, possivelmente, resultarem em uma entrada de usuário sem resposta, tornam a resposta do sistema mais coerente. Nesse caso, não teremos apenas resultados confirmando a veracidade ou contradizendo a afirmação apresentada. O sistema pode não ter informações em relação à sentença submetida pelo usuário.

Para o carregamento do modelo, é utilizada a biblioteca transformers, através do método `AutoModelForSequenceClassification.from_pretrained()`.

A função `run()` retorna as probabilidades de implicação, contradição e neutralidade, juntamente com cada sentença utilizada para calculá-las, e suas respectivas contribuições para o resultado final.

A integração dos módulos da busca semantica, Listing 5.4, e do módulo da inferência de linguagem natural, Listing 5.5, são feitos através de um pipeline de cosulta. Os pipelines de consulta são utilizados para receber uma consulta do usuário e gerar um resultado. Eles têm acesso a um DocumentStore que armazena um conjunto de documentos. Um pipeline de consulta padrão é projetado para retornar um resultado com base nos documentos armazenados no DocumentStore ao qual tem acesso. Nosso pipeline de consulta é composto de um `Retriever`, responsável por realizar a busca semântica, e do `EntailmentChecker`, responsável por mostrar o resultado agregado de implicação. Ele é definido como mostrado no Listing 5.6

Listing 5.6 – Pipeline de Consulta

```
pipe = Pipeline()
pipe.add_node(component=retriever, name="retriever", inputs=["Query"])
pipe.add_node(component=entailment_checker, name="ec", inputs=["
    retriever"])

params = {"retriever": {"top_k": 20}}
result = pipe.run(statements, params=params)
```

Para realizar a consulta e retornar os resultados basta rodar o método `run()` do pipeline que chama os métodos `run()` das duas classes em sequência. Para o retriever, é necessário fornecer o número de sentenças que ele ira buscar durante a busca semântica, no argumento `top_k`.

Enfim, projetamos e implementamos uma solução capaz de identificar sentenças falsas com base em um conjunto de sentenças verdadeiras. Associando técnicas de busca semântica por similaridade de embeddings com o Sentence-BERT e um modelo de Inferência de Linguagem Natural, conseguimos criar uma solução funcional para nosso problema. Agora, é apenas necessário testar o desempenho do nosso modelo e criar uma interface

visual para possibilitar uma interação fácil e eficiente com o usuário.

5.3 Interface Visual

A interface visual é construída utilizando o Streamlit, uma aplicação web para criar interfaces de usuário interativas baseadas em dados para visualização e interação com o usuário. Neste contexto, serão explicados os principais conceitos utilizados no código para criar a interface mostrada na Figura 7.

Figura 7 – Interface Visual (Fonte: Elaborado pelo Autor)

O Streamlit utiliza Markdown para a criação de conteúdo textual em seus aplicativos web. Markdown é uma linguagem de marcação leve que permite formatar textos de forma simples usando uma sintaxe fácil de ler e escrever. No Streamlit, Markdown é utilizado para adicionar títulos, parágrafos, listas, links e outros elementos textuais ao aplicativo.

Além disso, o Streamlit permite o armazenamento em cache de recursos como os modelos e a DocumentStore. Isso significa que esses elementos são carregados apenas na inicialização da aplicação. O decorador `@st.cache_resource` é utilizado nas funções responsáveis por inicializar a DocumentStore, o Retriever e o EntailmentChecker.

É possível armazenar dados persistentes entre pressionamentos de botões e alterações nos dados de entrada utilizando o atributo `st.session_state`. Assim, podemos

armazenar documentos e informações de implicação e contradição resultantes da consulta, além da própria entrada do usuário. Isso permite verificar mudanças no campo de texto em que o usuário insere a frase, possibilitando a redefinição das variáveis que carregam os resultados quando o usuário altera a sentença. Isso garante que os resultados anteriores não interfiram nos novos resultados.

O Streamlit reconhece objetos DataFrame da biblioteca pandas como uma tabela. Dessa forma, foi criado um DataFrame dos documentos com suas probabilidades de contradição, implicação e neutralidade para serem disponibilizados como fonte de informação consultada.

Assim, o usuário só precisa escrever a sentença que deseja verificar e clicar no botão **Run**. Então, as respostas agregadas e de cada documento aparecerão logo abaixo.

5.4 Testes e Avaliação

O primeiro passo para o teste do algoritmo era construir ou encontrar uma base de sentenças falsas e verdadeiras relacionadas à Amazônia Azul. Esse foi um grande desafio, já que informações estruturadas sobre a Amazônia Azul são escassas. Informações falsas estruturadas são ainda mais escassas. O método encontrado foi escrever manualmente um número significativo de frases para que pudéssemos testar o modelo. No entanto, poderíamos nos enviesar na escolha das frases para aquelas que já tínhamos conferido no banco de sentenças verdadeiras. A solução foi utilizar o ChatGPT (PENG et al., 2023), um chatbot online de inteligência artificial baseada em Transformers (GPT-3.5).

Neste processo, geramos 500 sentenças sobre a Amazônia Azul, sendo 250 verídicas e 250 falsas. Para garantir a coerência das informações obtidas, verificamos manualmente as frases geradas, para que pudéssemos ter um teste mais próximo da realidade. Após termos gerado as frases, inferimos a veracidade das sentenças obtidas através de nosso modelo.

Como nosso sistema depende da existência da informação verídica no banco de sentenças, algumas das frases não são encontradas. Isso pode ocorrer tanto porque o banco realmente não possui o conhecimento, quanto porque o conhecimento não existe por ser uma informação inventada, logo, falsa. Desse modo, podemos interpretar os resultados dos testes de duas formas. A primeira é desconsiderando as informações não encontradas, e a segunda é considerando-as como potencialmente falsas, já que se não existe no banco de sentenças verdadeiras sobre o assunto específico, ela possivelmente é inventada.

Para o primeiro teste que fizemos, escrevemos um prompt genérico, apenas solicitando que o GPT gerasse sentenças verdadeiras e falsas sobre o assunto. Para as sentenças verdadeiras ele funcionou adequadamente, entretanto para as falsas, as frases, apesar

de serem sobre a Amazônia Azul, eram, em grande parte, fantasiosas. Mesmo assim, calculamos o resultado. O resultado encontrado se encontra nas figuras 8, desconsiderando as informações não encontradas, e 9, considerando-as como falsas.

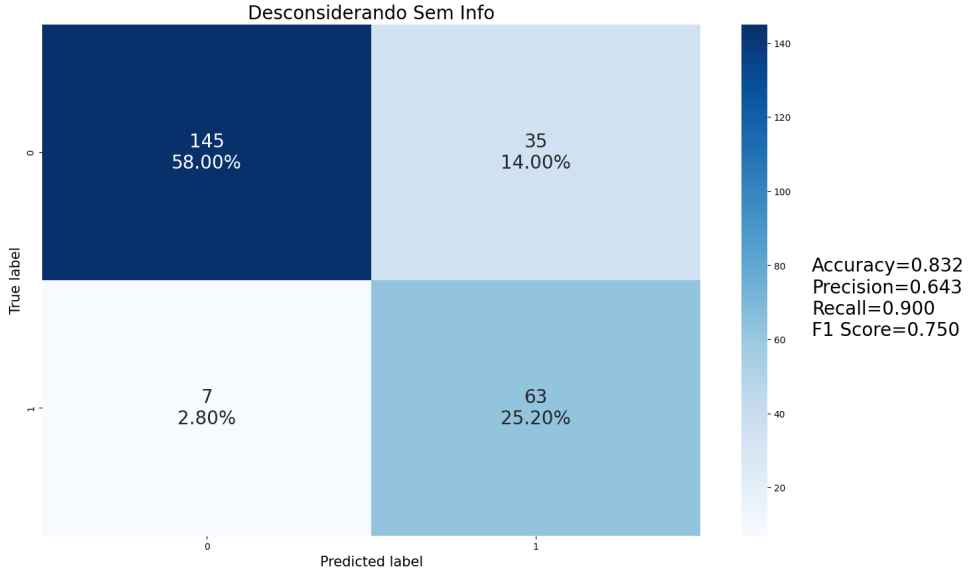

Figura 8 – Teste 1 - Informações não encontradas desconsideradas. (Fonte: Elaborado pelo Autor)

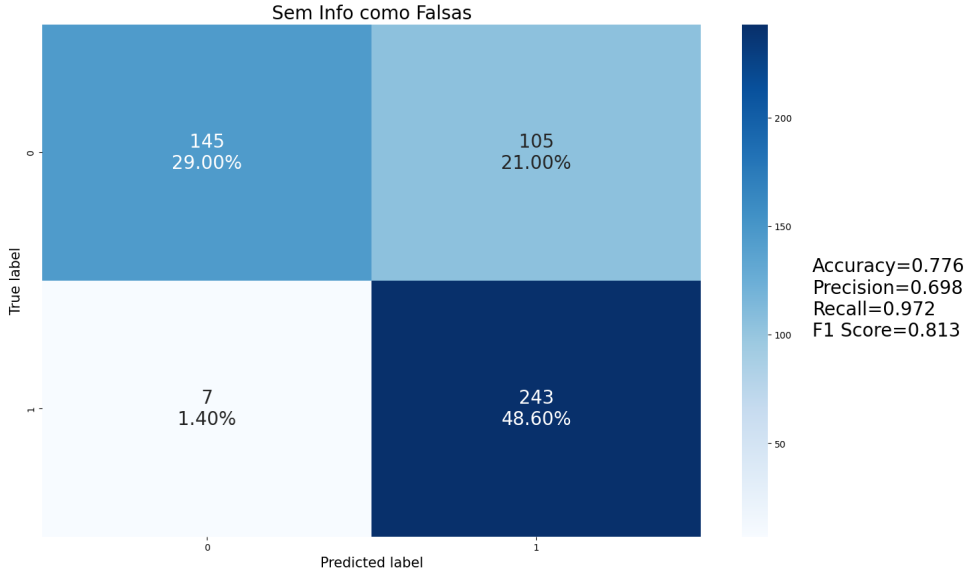

Figura 9 – Teste 1 - Informações não encontradas consideradas como falsas. (Fonte: Elaborado pelo Autor)

Nas figuras acima, o 0 (classe negativa) representa as sentenças verdadeiras, e o 1 (classe positiva) representa as sentenças falsas. Na figura 8, é mostrado o teste feito calculando as métricas apenas para as frases que foram encontradas.

Neste caso, o nosso modelo encontrou 50% das informações na base de sentenças verdadeiras (250 de 500 sentenças). Enquanto 72% das sentenças verdadeiras foram encontradas (180 de 250 sentenças), apenas 28% das falsas foram identificadas (70 de 250 sentenças). Vale reparar que isso ocorre devido às sentenças não realistas que o GPT gerou, indicando que, possivelmente, frases falsas tendem a ser menos identificadas do que frases verdadeiras.

Em relação às métricas, duas são as mais importantes: acurácia e sensibilidade (*recall*). A acurácia é a medida da proximidade de uma previsão ou medição em relação ao seu valor verdadeiro. Ou seja, a acurácia reflete a proporção de resultados verdadeiramente corretos (tanto verdadeiros positivos quanto verdadeiros negativos) entre todos os testes realizados. Os verdadeiros positivos são as sentenças falsas identificadas com precisão, enquanto os verdadeiros negativos são as sentenças verdadeiras igualmente bem identificadas.

A sensibilidade, também conhecida como *recall*, é a medida da capacidade de um teste ou modelo de identificar corretamente os casos positivos, no nosso caso, sentenças falsas. Em outras palavras, é a proporção de verdadeiros positivos em relação ao total de casos positivos. Um teste com alta sensibilidade raramente deixa passar casos positivos, mas pode ter mais falsos positivos.

Desconsiderando as informações não encontradas, conforme ilustrado na figura 8, alcançamos uma acurácia de 83,3% e uma sensibilidade de 90,3%. Já ao considerar tais informações como falsas, como mostrado na figura 9, a acurácia foi de 77,6% e a sensibilidade aumentou para 97,2%. Estes resultados são coerentes e indicam um bom desempenho do sistema, especialmente quando desconsideramos as sentenças não encontradas. Além disso, a alta sensibilidade do sistema é particularmente vantajosa, uma vez que notícias falsas interpretadas como verdadeiras são mais danosas do que notícias verdadeiras classificadas erroneamente como falsas.

Obtivemos um bom resultado ao desconsiderar as notícias não encontradas, o que indica a precisão do nosso sistema quando há informações disponíveis sobre o assunto. Com confiança na nossa base de dados, podemos considerar as notícias não encontradas como potencialmente falsas, especialmente porque foram menos frequentes do que as sentenças verdadeiras em situações envolvendo sentenças falsas fantasiosas.

Para diminuir a inventividade do GPT, realizamos um segundo teste com um prompt mais específico, solicitando que, para as frases falsas, fossem negadas informações verdadeiras, sem devaneios. Os resultados se encontram nas figuras 10 e 11.

Com o novo prompt, nosso modelo identificou 64,6% das sentenças (323 de 500). Entre as sentenças verdadeiras, 72% foram localizadas (180 de 250), e 57,2% das falsas foram reconhecidas (143 de 250). Embora o número de sentenças falsas identificadas tenha

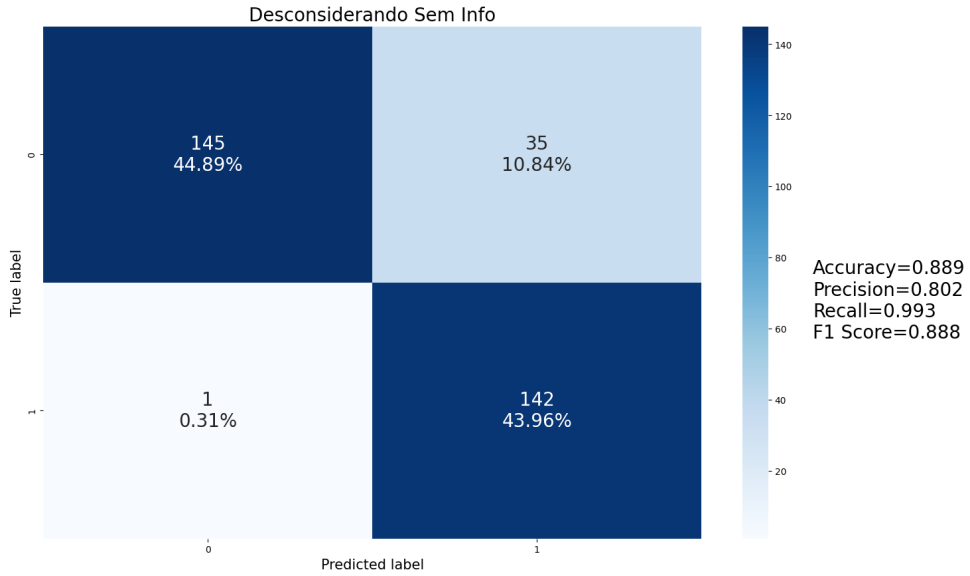

Figura 10 – Teste 2 - Informações não encontradas desconsideradas. (Fonte: Elaborado pelo Autor)

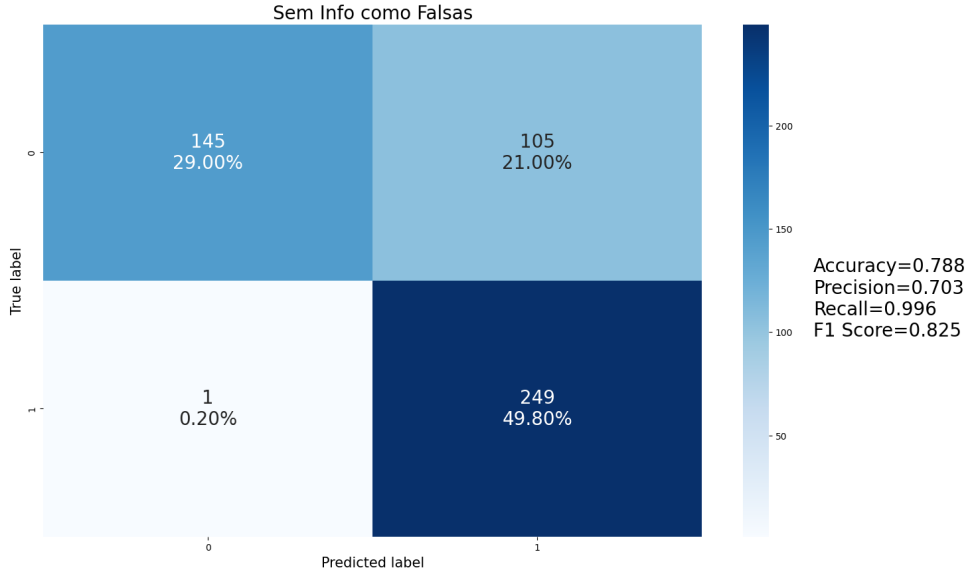

Figura 11 – Teste 2 - Informações não encontradas consideradas como falsas. (Fonte: Elaborado pelo Autor)

aumentado, ele ainda é menor que o número de sentenças verdadeiras identificadas. Isso indica o padrão observado anteriormente, onde sentenças semelhantes às falsas são mais dificilmente encontradas no banco de dados de sentenças verdadeiras. Além disso, este teste está mais alinhado com a realidade, já que as informações falsas tendem a ser muito parecidas com as verossímeis.

Desconsiderando as informações não encontradas, conforme mostrado na figura

10, alcançamos uma acurácia de 88,9% e uma sensibilidade de 99,3%. Considerando tais informações como falsas, conforme a figura 11 ilustra, a acurácia foi de 78,8% e a sensibilidade, de 99,6%. Observamos resultados semelhantes nos dois cenários, porém, indicando uma resposta mais concreta para mais sentenças, sugerindo que elas poderiam ser possivelmente falsas.

Se conseguíssemos aumentar o banco de dados de sentenças verdadeiras e aprimorar o processo de tratamento e verificação das sentenças em relação à fonte, teríamos maior confiança ao classificar sentenças não encontradas como potencialmente falsas.

6 Considerações Finais

6.1 Conclusões do Projeto de Formatura

Esta seção apresenta um balanço do trabalho realizado, destacando os resultados alcançados e não alcançados, juntamente com as justificativas pertinentes.

Considerando o teste mais coerente, identificado nas figuras 10 e 11, foram obtidas uma acurácia de 88,9% e uma sensibilidade de 99,3%, desconsiderando informações não encontradas, e uma acurácia de 78,8% e uma sensibilidade de 99,6%, considerando-as como falsas.

Durante o projeto, enfrentamos diversas dificuldades, tais como a falta de bases estruturadas sobre o tema, limitações computacionais para o cálculo dos embeddings e a ausência de ajustes nos modelos pré-treinados utilizados. A principal dificuldade esteve relacionada à falta de uma base de dados adequada, o que impossibilitou a criação de um sistema mais preciso devido à má formatação dos textos extraídos. Além disso, inviabilizou a realização de ajustes no modelo com dados específicos do nosso escopo, dificultando a execução de testes mais coerentes.

Apesar das dificuldades, conseguimos desenvolver um sistema funcional, com uma interface visual que permite a interação dos usuários com o trabalho. Além disso, o sistema obteve um desempenho satisfatório, tanto considerando frases não encontradas como falsas, quanto desconsiderando-as.

Adicionalmente, o aprendizado profundo sobre Processamento de Linguagem Natural adquirido durante o trabalho foi enriquecedor, assim como a aplicação dos conceitos de Inteligência Artificial e desenvolvimento de sistemas aprendidos ao longo do curso.

6.2 Contribuições

A contribuição da equipe consistiu em integrar dois algoritmos de Processamento de Linguagem Natural para desenvolver um sistema capaz de identificar sentenças falsas em um escopo específico. Conforme mencionado na seção 4.3.3, os modelos de verificação de notícias falsas baseados em bancos de conhecimento são escassos. Apesar das adversidades, o grupo conseguiu cumprir satisfatoriamente esse desafio.

Além disso, a escolha de trabalhar com o escopo específico da Amazônia Azul é uma grande contribuição, considerando a importância dessa área para o desenvolvimento econômico nacional. Vale ressaltar que, ao possuir informações sobre um escopo específico,

o projeto demonstra flexibilidade para adaptar-se a outros temas. Dessa forma, uma contribuição adicional é a capacidade do sistema de se ajustar a diferentes escopos.

Por último, as bases adquiridas e construídas através do Google Scholar podem auxiliar outros grupos de pesquisa a incrementar suas bases para obterem mais informações, passando por um processo de checagem. Obtendo um modelo mais preciso com modificações futuras, é possível também auxiliar e acelerar no processo de verificação de sentenças para a construção de WIKIs confiáveis, como a BLAB-wiki.

6.3 Perspectivas de Continuidade

Existem diversas formas de incrementar esse projeto, tanto melhorando sua funcionalidade quanto sua precisão.

Começando pela utilização do usuário, associado a sistemas de busca na internet e fontes, os links das informações obtidas poderiam ser fornecidos junto com o resultado.

Além disso, quanto ao sistema, diversos avanços podem ser feitos para melhorar a precisão do algoritmo. A primeira seria uma inicial construção de uma base estruturada, para ajuste dos modelos de similaridade de sentenças e de inferência de linguagem natural. Dentro de um domínio específico, o modelo seria capaz de entender melhor as nuances, comparações e construções linguísticas relacionados a este tema.

Adicionalmente, testes poderiam ser feitos de uma forma mais concisa, possibilitando um melhor acompanhamento de métricas para alterar os hiperparâmetros do modelo, como os thresholds de consideração de similaridade e de probabilidade de implicação ou contradição.

Ademais, poderíamos ajustar os modelos de similaridade textual e de inferência de linguagem natural para que fossem mais atrelados ao domínio. Reconhecendo melhor a relação semântica entre as palavras em um domínio especifico.

Por fim, obtendo uma base maior e melhor estruturada, teríamos uma fonte mais confiável de informação, obtendo uma cobertura de informação maior. Além do mais, poderia ser criado um modelo de geração de novas informações, a partir de informações existentes, combinando informações de diferentes fontes.

Uma explicação final poderia ser fornecida ao usuário utilizando algum modelo de geração de texto, que explicasse a relação da entrada do usuário com o resultado obtido, dando mais confiança ao uso do sistema.

Referências

BOWMAN, S. R. et al. A large annotated corpus for learning natural language inference. In: MÀRQUEZ, L.; CALLISON-BURCH, C.; SU, J. (Ed.). *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 632–642. Disponível em: <https://aclanthology.org/D15-1075>. Citado 2 vezes nas páginas 24 e 41.

CONNEAU, A. et al. Xnli: Evaluating cross-lingual sentence representations. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. [S.l.]: Association for Computational Linguistics, 2018. Citado na página 24.

DAGAN, I.; GLICKMAN, O.; MAGNINI, B. The pascal recognising textual entailment challenge. In: QUIÑONERO-CANDELA, J. et al. (Ed.). *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 177–190. ISBN 978-3-540-33428-6. Citado na página 23.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. Citado 6 vezes nas páginas 7, 14, 15, 18, 19 e 21.

DHINAKARAN, A. *Getting Started With Embeddings Is Easier Than You Think*. 2022 (Acesso em 5 mar. 2023). Disponível em: <https://towardsdatascience.com/getting-started-with-embeddings-is-easier-than-you-think-e88b7b10bed1>. Citado 2 vezes nas páginas 7 e 19.

HE, P.; GAO, J.; CHEN, W. *DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. 2023. Citado na página 24.

HOWARD, J.; RUDER, S. *Universal Language Model Fine-tuning for Text Classification*. 2018. Citado na página 24.

JOHNSON, J.; DOUZE, M.; JÉGOU, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, IEEE, v. 7, n. 3, p. 535–547, 2019. Citado na página 37.

JR., A. C. R. The conference on mechanical translation. *Mech. Transl. Comput. Linguistics*, v. 1, n. 3, p. 47–55, 1954. Disponível em: <http://www.mt-archive.info/MT-1954-Reynolds.pdf>. Citado na página 18.

KALIYAR, R. K.; GOSWAMI, A.; NARANG, P. Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia tools and applications*, Springer, v. 80, n. 8, p. 11765–11788, 2021. Citado na página 15.

KUMAR, S.; SHAH, N. *False Information on Web and Social Media: A Survey*. 2018. Citado na página 14.

LAURER, M. et al. *Less Annotating, More Classifying – Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and BERT-NLI*. OSF, 2022. Disponível em: <osf.io/wqc86>. Citado 2 vezes nas páginas 24 e 42.

MIKOLOV, T. et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. Citado na página 18.

PASCHOAL, A. F. et al. Pirá: a bilingual portuguese-english dataset for question-answering about the ocean. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. [S.l.: s.n.], 2021. p. 4544–4553. Citado na página 13.

PENG, A. et al. Gpt-3.5 turbo fine-tuning and api updates. *OpenAI Blog*, Aug 2023. Disponível em: <https://openai.com/blog/gpt-3-5-turbo/>. Citado 2 vezes nas páginas 18 e 46.

PEREIRA, R. O que é a amazônia azul e por que o brasil quer se tornar potência militar no atlântico. *Marinha do Brasil*, 11 2019. Disponível em: <https://www.marinha.mil.br/economia-azul/noticias/o-que-%C3%A9-amaz%C3%B4nia-azul-e-por-que-o-brasil-quer-se-tornar-pot%C3%AAncia-militar-no-atl%C3%A2ntico>. Citado na página 17.

PIROZELLI, P. et al. *The BLue Amazon Brain (BLAB): A Modular Architecture of Services about the Brazilian Maritime Territory*. 2022. Citado 2 vezes nas páginas 15 e 32.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. Citado 4 vezes nas páginas 7, 19, 22 e 23.

RICHARDSON, L. *Beautiful soup documentation*. [S.l.]: April, 2007. Citado na página 29.

TURING, A. Computing machinery and intelligence. *Mind*, v. 59, n. 236, p. 433–460, 1950. Citado na página 18.

VASWANI, A. et al. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017. Citado 6 vezes nas páginas 7, 14, 18, 19, 20 e 21.

VICARIO, M. D. et al. Polarization and fake news: Early warning of potential misinformation targets. *ACM Transactions on the Web (TWEB)*, ACM New York, NY, USA, v. 13, n. 2, p. 1–22, 2019. Citado na página 13.

VOSOUGHI, S.; ROY, D.; ARAL, S. The spread of true and false news online. *science*, American Association for the Advancement of Science, v. 359, n. 6380, p. 1146–1151, 2018. Citado na página 13.

YANG, Y. et al. *Multilingual Universal Sentence Encoder for Semantic Retrieval*. 2019. Citado na página 41.