CENTRO UNIVERSTIÁRIO SALESIANO DE SÃO PAULO

UNIDADE LORENA

CURSO DE ENGENHARIA DE COMPUTAÇÃO

Daniel Lucas Pessoa Monteiro dos Santos

Fernando Marcelinho Martins

Victor Aarão Lemes

Vitor Emanuel Vieira da Silva

PROJETO INTEGRADOR:

FakeNo

Lorena
2023

RESUMO

O presente trabalho descreve o processo de desenvolvimento da extensão FakeNo*,* para *Google Chrome*, a qual permite seus usuários verificar a veracidade de artigos de sites de notícias com um clique. A extensão utiliza de técnicas de processamento de linguagem natural (PLN), inteligência artificial (IA) e *web scrapping* para classificar notícias como reais ou falaciosas, provendo aos usuários uma ferramenta confiável para avaliar a credibilidade de informações online.

Palavras-chave: Fake News; inteligência artificial; *web scrapping*.

SUMÁRIO

1. INTRODUÇÃO

Com o avanço da tecnologia e o fácil acesso a informações online, a disseminação de notícias falsas (*fake News*) se tornou uma preocupação significativa. É considerada notícia falsa toda forma de desinformação distribuída a partir de algum meio de comunicação, podendo ser jornal impresso, rádio, televisão e afins. Notícias falsas costumam apelar para o emocional, muitas vezes misturando acontecimentos reais com inverdades, buscando uma maior adesão e mais compartilhamentos. No Brasil, quatro a cada dez pessoas afirmam receber notícias falsas todos os dias (GUIMARÃES; RODRIGUES, 2022)

Diante desse cenário, o presente projeto propõe o desenvolvimento e a implementação da extensão FakeNo para o navegador Google Chrome, com o objetivo de auxiliar os usuários na verificação da veracidade de notícias. Através da aplicação de técnicas de inteligência artificial e web scraping, a extensão oferece uma solução prática e acessível para avaliar a credibilidade das informações encontradas na internet.

Com a dificuldade de encontrar uma ferramenta para facilitar o compreendimento da verdade, este projeto visa aprimorar a capacidade dos usuários de identificar notícias falsas, com o intuito de proporcionar uma experiência mais segura e confiável durante a navegação na web. Para isso, pretende-se desenvolver um modelo de inteligência artificial capaz de classificar notícias como verdadeiras ou falsas, utilizando técnicas de *machine learning*. Além disso, busca-se implementar a extensão FakeNo, que permitirá aos usuários obter um feedback instantâneo sobre a confiabilidade de uma notícia ao visitar um site de notícias.

A importância deste projeto se dá por conta da contribuição proporcionada para combater a disseminação de informações enganosas e promover a conscientização sobre a importância da verificação de notícias. Com o aumento do consumo de informações online, torna-se crucial fornecer aos usuários ferramentas que os ajudem a discernir entre notícias verdadeiras e falsas. A extensão FakeNo, ao oferecer uma verificação rápida e fácil, possibilita uma navegação mais segura e confiável, evitando que usuários

sejam influenciados ou prejudicados por informações imprecisas ou enganosas.

Espera-se que, ao final deste projeto, a extensão FakeNo esteja plenamente funcional e disponível para uso pelos usuários do Google Chrome. Com a implementação bem-sucedida da inteligência artificial e das técnicas de web scraping, espera-se alcançar uma alta precisão na classificação das notícias, proporcionando aos usuários uma ferramenta confiável para verificar a veracidade das informações encontradas online. Além disso, espera-se que a extensão FakeNo contribua para aumentar a conscientização sobre a disseminação de notícias falsas e ajude a construir uma cultura de checagem de fatos mais sólida entre os usuários da internet.

1.1. OBJETIVOS

Este projeto tem como objetivo o desenvolvimento de uma extensão para o navegador Google Chrome, que permite que usuários verifiquem a veracidade de notícias online com apenas um clique.

2. METODOLOGIA

A primeira abordagem pensada para esse trabalho foi a criação de um modelo de inteligência artificial que pudesse classificar notícias como verdadeiras ou falsas. Após o desenvolvimento, a ideia era utilizá-la junto a uma extensão de navegador para, assim, ao acessar um site de notícias, o usuário ter um feedback sobre a veracidade da notícia.

Para criar uma IA – Inteligência artificial, que possa classificar notícias como verdadeiras ou falsas, foi conduzida uma pesquisa na internet em diversas fontes através do buscador Google e da inteligência artificial *chatGPT*, para que fosse compreendido como seria feita a criação da IA. Com base nessas pesquisas chegou-se à conclusão de que o algoritmo de *Logistic Regression* seria a opção a ser utilizada, pelo fato de ser utilizado para encontrar relações entre duas ou mais variáveis além de fazer classificação binária.

Ficou entendido também a não possibilidade de inserir textos diretamente no algoritmo de *machine learning,* sem fazer o devido tratamento

antes. Para tratar o texto, fez-se necessário primariamente remover palavras de ligação, como preposições, artigos etc. e caracteres especiais, além de reduzir palavras ao seu radical e o mais importante: transformá-las em números usando de um algoritmo de vetorização.

O algoritmo de vetorização escolhido foi o *TfidfVectorizer*, que atribui números para cada palavra com base não somente na contagem de palavras, mas também na frequência que essas aparecem nos textos.

Com esse entendimento, foi necessário procurar um *dataset*, conjunto de dados, para ser utilizado no treinamento do modelo. O conjunto encontrado foi o *FakeRecogna*, que conta com um total de 11902 notícias de diversas categorias, classificadas como verdadeiras ou falsas com base em agências de checagem de fatos.

Para a modelagem, foram utilizadas as seguintes bibliotecas: *pandas*, para manipular o *dataset*; *re*, para remover os caracteres especiais do texto; *nltk*, para eliminar palavras de ligação e reduzir as palavras aos seus radicais. Por fim, a biblioteca *sklearn* foi utilizada para realizar o treinamento e vetorização, utilizando os algoritmos mencionados anteriormente.

Após algumas tentativas de modelagem da IA que resultaram em modelos com baixa acurácia, uma ótima referência foi encontrada para a criação do FakeNo-AI, modelo final de IA deste trabalho. Essa referência foi identificada em um vídeo do canal do Youtube *Codifike*, apresentando uma acurácia de 96%.

Para disponibilizar de maneira simples a uso da IA, seu modelo e vetorizador foram exportados para serem utilizados em uma API RESTful feita com *python* e *Flask*. *Flask* é um *micro-framework* web para *pyhton*, o qual foi escolhido graças sua facilidade de desenvolvimento.

Para que a notícia seja classificada, primeiro é necessário ter acesso às informações dela, como título, autor(es) e um resumo de seu conteúdo. Para fazer a raspagem dessas informações da internet (*web scrapping*), bem como o resumo do conteúdo da notícia, utilizou-se da biblioteca *newspaper3k* do python.

Com a primeira versão da API já em funcionamento, foi dado início ao desenvolvimento da extensão do Google Chrome. Essa extensão, ao carregar de uma página no navegador, verifica se o conteúdo é um artigo por meio de uma meta *tag* e, em seguida, executa um script em JavaScript dentro da página.

O script capta a URL da página aberta e envia para a API, a qual faz a raspagem (web scrapping), e classificação da notícia com a FakeNo-AI, assim, retornando com a confirmação da notícia, informando ser verídica ou falaciosa. Com base nesse resultado, um ícone de alerta ou de verificado é exibido ao lado do título da notícia, como podem ser observados nas figuras 1 e 2.

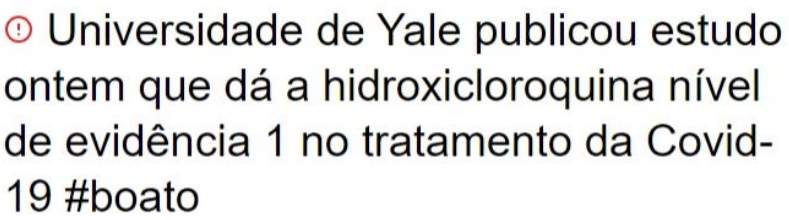

Figura 1: Exemplo de notícia falsa.

Figura 2: Exemplo de notícia verdadeira.

3. RESULTADOS E DISCUSSÃO

Com a primeira versão já em funcionamento, foram observados alguns pontos de melhoria. Foi identificado que o usuário recebia notícias classificadas, independentemente de sua preferência, o que resultava em chamadas desnecessárias à API. Além disso, constatou-se que um ícone indicando a veracidade da notícia não era suficiente para fornecer um feedback adequado ao usuário. Foi necessário então fornecer um feedback mais preciso para que o usuário pudesse determinar se uma notícia era confiável ou não. Com base nos pontos mencionados o desenvolvimento da segunda versão da aplicação foi iniciado.

Para proporcionar ao usuário um feedback satisfatório, a decisão foi de utilizar a API da OpenAI, especificamente o modelo de linguagem natural GPT (*Generative Pre-trained Transformer*). Esse modelo é empregado para realizar a análise da veracidade de notícias, determinando como verdadeiras ou falsas, além de gerar um parágrafo explicativo que justifica a conclusão obtida.

Para permitir que o GPT chegue a uma conclusão mais precisa e gere um parágrafo em um formato coerente, uma instrução é escrita, especificando que dados serão informados e qual deve ser o formato de resposta. Os dados informados consistem nas informações da notícia que deve ser classificada, a classificação feita pela FakeNo-AI e as informações dos três primeiros resultados ao buscar sobre o assunto da notícia no google.

Porém, antes do parágrafo de conclusão ser gerado são necessários alguns passos. Na primeira versão da API, o processo envolvia buscar os dados da notícia, classificá-la utilizando o FakeNo-AI e retornar o resultado. Essas etapas ainda se mantém, entretanto, com alguns passos a mais.

Após a busca dos dados da notícia, o GPT foi utilizado para extrair o assunto e criar um texto para ser buscado no Google. A busca é feita com o auxílio da biblioteca *googleapiclient,* para fazer o uso da *Custom Search API* da Google.

Com as URLs dos três primeiros resultados faz-se o uso do *ThreadPoolExecutor,* do python, para realizar a raspagem dos dados dessas páginas em paralelo, a fim de evitar demora na resposta do usuário.

As informações extraídas das buscas feitas no Google são formatadas, para que possam ser utilizadas na instrução dada ao GPT posteriormente. Então é feita a classificação pela FakeNo-AI e é montado o texto que é enviado para o GPT chegar à conclusão sobre a veracidade da notícia.

A rota de análise da API ficou mais completa, mas também mais demorada. Neste contexto, fazer com que essa rota retornasse uma stream da dados foi necessário. A cada passo concluído no processo de análise da notícia é retornado um JSON com o nome da etapa e seu resultado. Dessa forma é possível dar um feedback de status para o usuário.

Com todas as alterações na API feitas, entre outras refatorações para melhorar a qualidade do código, foi necessário atualizar a extensão.

Com o aumento da complexidade, foi decidido mover o código a ser inserido na página, tanto o Javascript como CSS, para arquivos próprios disponibilizados pela API em *Flask*.

A extensão, depois de atualizada, apenas verifica se a página é um artigo e, se for, adiciona uma *tag* script com o link para o Javascript disponibilizado pela API e um link para o CSS.

O script, por sua vez, adiciona um botão cinza com um ícone de lupa na cor branca junto do título do artigo da página. Então, ao receber o evento de *hover,* exibe um *tooltip* com a palavra “analisar”.

Figura 3: Representação da atualização feita.

A chamada a API é feita após o clique no botão. Em seguida, o botão exibe um *spinner* para indicar o carregamento, e ao receber o evento de *hover* é possível acompanhar o progresso da análise da notícia. Ao receber o resultado de cada passo, sinaliza que o processo foi concluído.

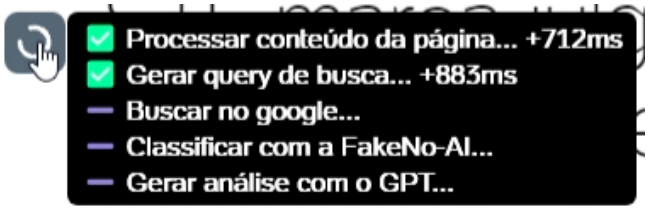

Figura 4: Indicação das etapas do processo.

Quando a análise é finalizada, o botão é desabilitado. Abaixo do título da página surge, com uma animação suave, um *card* com bordas e cabeçalho verde se a notícia for verdadeira, ou então, bordas e cabeçalho vermelho se falsa. Dentro desse card, o parágrafo criado pelo GPT descreve a razão pela qual a notícia é classificada como verídica ou falaciosa, e ainda aponta as referências utilizadas para chegar nessa conclusão. Abaixo, seguem dois exemplos:

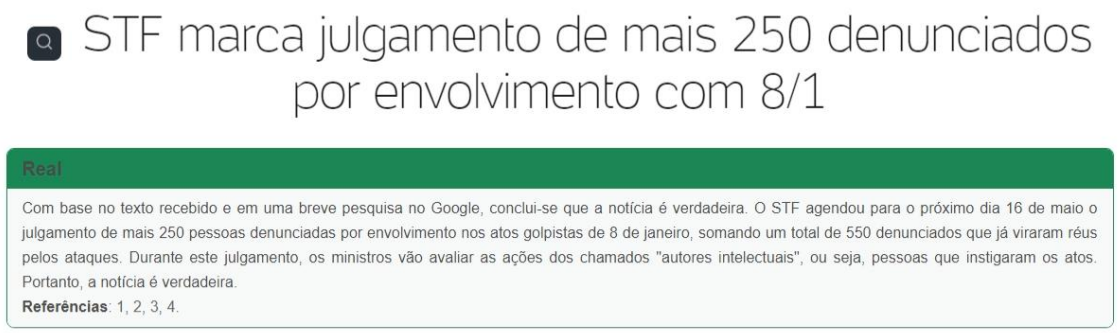

Figura 5: Exemplo de notícia verdadeira.

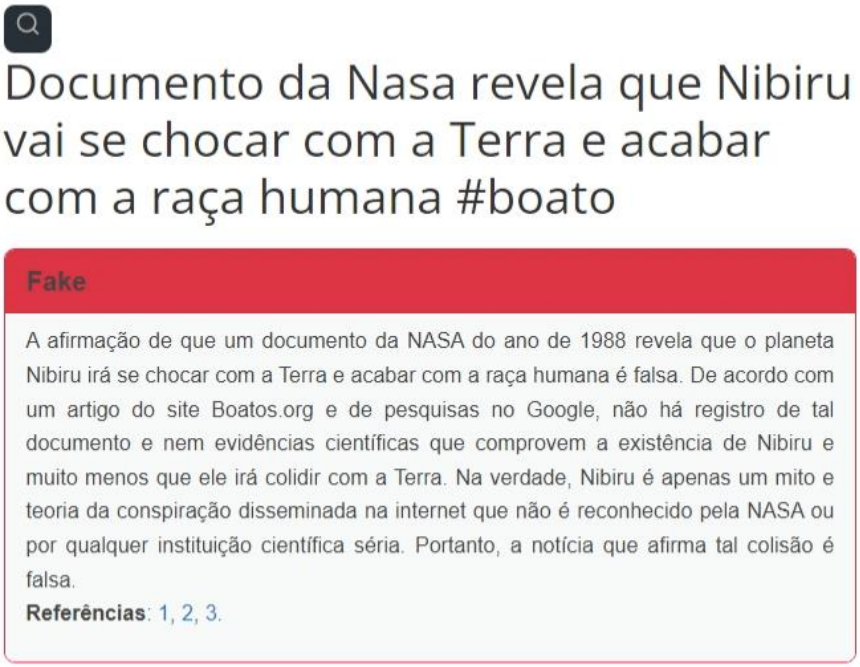

Figura 6: Exemplo de notícia falsa.

4. CONCLUSÃO

Em conclusão, o desenvolvimento e a implementação da extensão FakeNo para o navegador Google Chrome representa uma solução prática e acessível para auxiliar os usuários na verificação da veracidade das notícias encontradas na internet. Através da aplicação de técnicas de inteligência artificial e web scraping, a extensão oferece aos usuários um meio eficiente de avaliar a credibilidade das informações, ajudando a combater a disseminação de notícias falsas.

5. REFERÊNCIAS

CODIFIKE. Machine Learning para prever fake news usando python #machineleaming. 2021. Disponível em: <**Machine Learning para prever fake news usando python #machinelearning - YouTube**> Acesso em: 04/06/2023

GARCIA, Gabriel Lino. FakeRecogna. 2022. Disponível em: < **datasets/FakeRecogna at master · recogna-lab/datasets · GitHub**> Acesso em: 04/06/2023

GOOGLE WORKSPACE. Guia de início rápido do Python. 2023. Disponível em: <**https://developers.google.com/docs/api/quickstart/python?hl=pt-br**> Acesso em: 04/06/2023

GUIMARÃES, Pedro; RODRIGUES, Cleber. 4 em cada 10 brasileiros afirmam receber fake News diariamente. Rio de Janeiro, 2022. Disponível em: < **4 em cada 10 brasileiros afirmam receber fake news diariamente (cnnbrasil.com.br)**> Acesso em: 04/06/2023

NLTK. Sample usage for portuguese_en. 2023. Disponpivel em: <**NLTK :: Sample usage for portuguese_en**> Acesso em: 04/06/2023

NumFOCUS, Inc. pandas. 2023. Disponível em: <**https://pandas.pydata.org**> Acesso em: 04/06/2023

OPENAI. Introduction – OpenAI API. 2023. Disponível em: < **https://platform.openai.com/docs/introduction/overview**> Acesso em: 04/06/2023

OU-YANG, Lucas. Newspaper3k: Article scraping & curation. 2013. Disponível em: <**Newspaper3k: Article scraping & curation — newspaper 0.0.2 documentation**> Acesso em: 04/06/2023

PALLETS. Flask. 2010. Disponível em: <**https://flask.palletsprojects.com/en/2.3.x/**> Acesso em: 04/06/2023

PEDREGOSA et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011. Disponível em: <**https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html**> Acesso em: 04/06/2023

PEDREGOSA et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011. Disponível em: <**https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression>** Acesso em: 04/06/2023

PEDREGOSA et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011. Disponível em: <**https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html**> Acesso em: 04/06/2023

PEDREGOSA et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011. Disponível em: <**https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction**> Acesso em: 04/06/2023

PEDREGOSA et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011. Disponível em: <**https://scikit-learn.org/stable**> Acesso em: 04/06/2023

PROGRAMMABLE SEARCH ENGINE. JSON API de pesquisa personalizada. 2023. Disponpivel em: <**https://developers.google.com/custom-search/v1/overview?hl=pt-br**> Acesso em: 04/06/2023

PYTHON SOFTWARE FOUNDATION. Library. 2023. Disponível em: <**re — Regular expression operations — Python 3.11.3 documentation**> Acesso em: 04/06/2023