UNIVERSIDADE ESTADUAL PAULISTA JÚLIO MESQUITA FILHO
INSTITUTO DE CIÊNCIA E TECNOLOGIA DE SOROCABA

BRUNO LOBO DE MELO

ANÁLISE DE SENTIMENTOS: USANDO A INTELIGÊNCIA ARTIFICIAL PARA COMBATER COMENTÁRIOS INDESEJÁVEIS

Sorocaba

Dezembro de 2022

BRUNO LOBO DE MELO

ANÁLISE DE SENTIMENTOS: USANDO A INTELIGÊNCIA ARTIFICIAL PARA COMBATER COMENTÁRIOS INDESEJÁVEIS

Trabalho de Conclusão de Curso apresentado junto ao Curso de Engenharia de Controle e Automação Bacharel do Universidade Estadual Paulista Júlio Mesquita Filho - Instituto de Ciência e Tecnologia de Sorocaba, como requisito parcial à obtenção do título de Engenheiro de Controle e Automação.

Orientador:

Prof. Dr. Ivando Severino Diniz

Sorocaba

Dezembro de 2022

M528a

Melo, Bruno Lobo de

Análise de Sentimentos : Usando a inteligência artificial para combater comentários indesejados / Bruno Lobo de Melo. -- Sorocaba, 2022

58 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba

Orientador: Ivando Severino Diniz

1. API. 2. Meta. 3. Análise de sentimentos. 4. Naive Bayes. 5. Inteligência artificial. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia de Sorocaba

ANÁLISE DE SENTIMENTOS: USANDO A INTELIGÊNCIA ARTIFICIAL PARA COMBATER COMENTÁRIOS INDESEJÁVEIS

Bruno Lobo de Melo

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE
BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Prof. Dr. Everson Martins
Coordenador

BANCA EXAMINADORA:

Prof. Dr. Ivando Severino Diniz
Orientador / UNESP – Campus Sorocaba

Prof. Dr. Ivando Severino Diniz
Orientador / UNESP – Campus Sorocaba

Eng. Diego Mariano
UNESP – Campus de Sorocaba

Dezembro 2022

Dedico este trabalho primeiramente à minha noiva, Yasmin Soares Marins, por ter se mantido ao meu lado durante todo o meu percurso, e a minha família, por sua paciência e aparo.

AGRADECIMENTOS

Ao meu professor, Dr. Ivando tanto pela orientação desse trabalho, quanto pelas aulas ministradas durante o curso.

Aos meus colegas e amigos que tornaram essa jornada ainda mais prazerosa e contribuíram para minha formação, como pessoa e engenheiro.

“Nossa maior fraqueza é a desistência. O caminho mais certeiro para o sucesso é sempre tentar apenas uma vez mais.”

Thomas Edison

RESUMO

Nesse trabalho foi desenvolvido uma ferramenta que permite ao usuário conectar a sua rede social, citada o Instagram, e realizar a análise dos comentários relacionados as postagens da conta. Classificando-as nas notas de entre positivas e negativas. Esse processo foi dividido em etapas, sendo a primeira etapa o desenvolvimento de um método para coletar os comentários de um conta escolhida dentro da rede social, isso pode ser realizado com a ajuda de uma API. Com os dados coletados, foi treinado um modelo de naive bayes para classificar os dados conforme proposto, em positivos e negativos, e submeter os dados coletados a esse classificador, para os dados classificados como negativos, seria necessário deletá-los da plataforma, que também é possível essa realização via API. Para o modelo treinado foi alcançado uma precisão no julgamento dos comentários de acima de 80%, ajudando assim eliminar muitos dos comentários não desejados de uma rede social.

Palavras-chave: TCC, TG, Instagram, Meta, API, Análise de Sentimentos, Naive Bayes.

ABSTRACT

In this paper, a tool was developed that allows the user to connect to their social network, mentioned Instagram, and perform the analysis of comments related to the account's posts. Classifying them in the notes between positive and negative. This process was divided into stages, with the first stage being the development of a method to collect the comments from a chosen account within the social network, which can be done with the help of an API. With the collected data, a naive bayes model was trained to classify the data as proposed, in positive and negative, and submit the collected data to this classifier, for data classified as negative, it would be necessary to delete them from the platform, which is also possible to do via API. For the trained model, an accuracy in the judgment of the comments of over 80% was achieved, thus helping to eliminate many of the unwanted comments from a social network.

Keywords: Instagram, Meta, API, Sentimental Analysis, Naive Bayes.

SUMÁRIO

1 OBJETIVOS. 9

2 INTRODUÇÃO 10

3 REVISÃO BIBLIOGRÁFICA 13

3.1 Python para análise de sentimentos 13

3.1.1 NLTK (Natural Language Toolkit) 13

3.2 Utilização de APIs para extração de dados em redes sociais 14

4 PROPOSIÇÃO 17

5 METODOLOGIA 19

5.1 Aquisição de dados 19

5.1.1 Python 19

5.1.2 Bibliotecas 21

5.1.3 Meta API 22

5.1.4 Jupyter 23

5.2 Tratamento de dados 24

5.3 Naive Bayes 26

5.4 Webhook 27

6.1 Aquisição de dados 29

6.1.1 Chamadas da API 29

6.1.2 Extração dos comentários. 29

6.2 Análise de dados 30

7 CONCLUSÃO 32

8 REFERÊNCIAS BIBLIOGRÁFICAS 33

ANEXO A – Autenticação na API 34

ANEXO B – Requisição de comentários 36

ANEXO C – Análise dos dados 51

1 OBJETIVOS.

O objetivo deste trabalho de graduação compreende no estudo e desenvolvimento de uma solução que permitam que empresas monitorem suas redes sociais e filtrem, de modo automático os comentários desejados em suas plataformas.

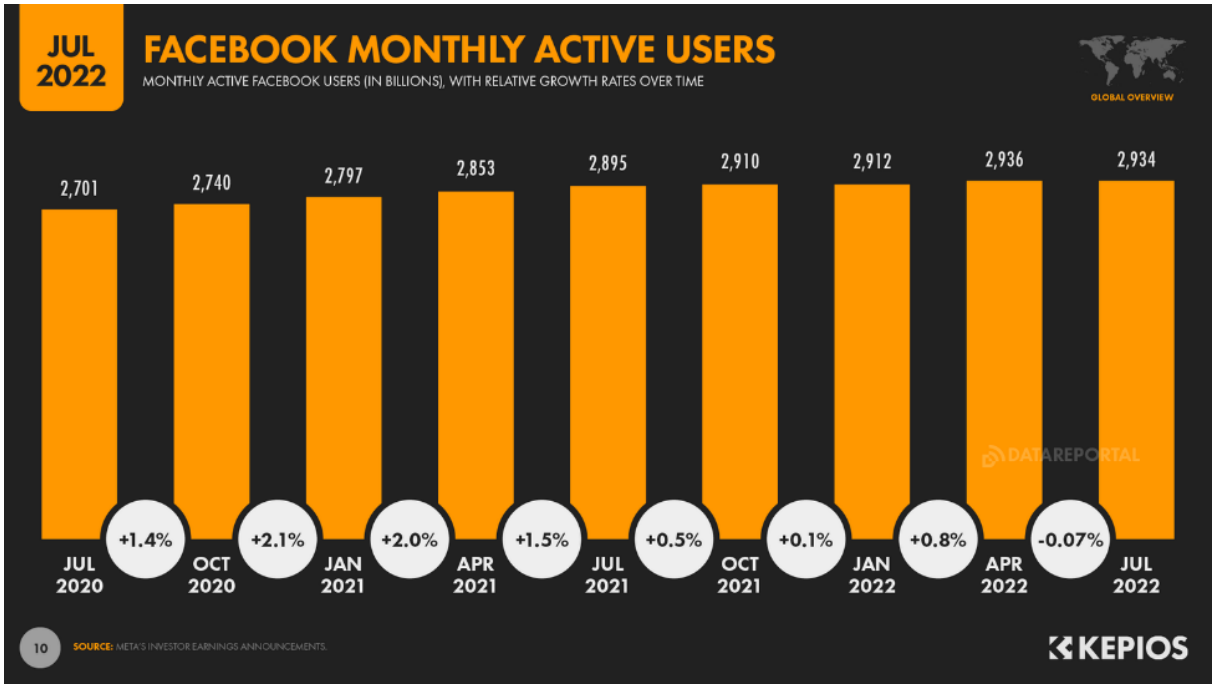
Para que isso seja feito de modo efetivo o trabalho será dividido em 4 etapas.

- Criação de um agente, capaz de acessar os comentários da rede social, por meio de uma API (Interface de programação de aplicações).
- Classificação dos comentários já existentes entre desejáveis e indesejáveis.
- Atuação do agente nos comentários já existentes.
- Monitoramento de novos comentários por meio de uma aplicação web.

2 INTRODUÇÃO

Desde a sua fundação em 2004, o Facebook tem experimentado um crescimento constante no número de usuários de sua rede social. De acordo com o relatório apresentado aos investidores da empresa em 2022, a base de usuários do Facebook atingiu cerca de 2,9 bilhões (ver Figura 1), o que representa aproximadamente 36% da população mundial. O crescimento exponencial das redes sociais tem sido notável nos últimos anos e o Facebook é um dos principais players nesse mercado.

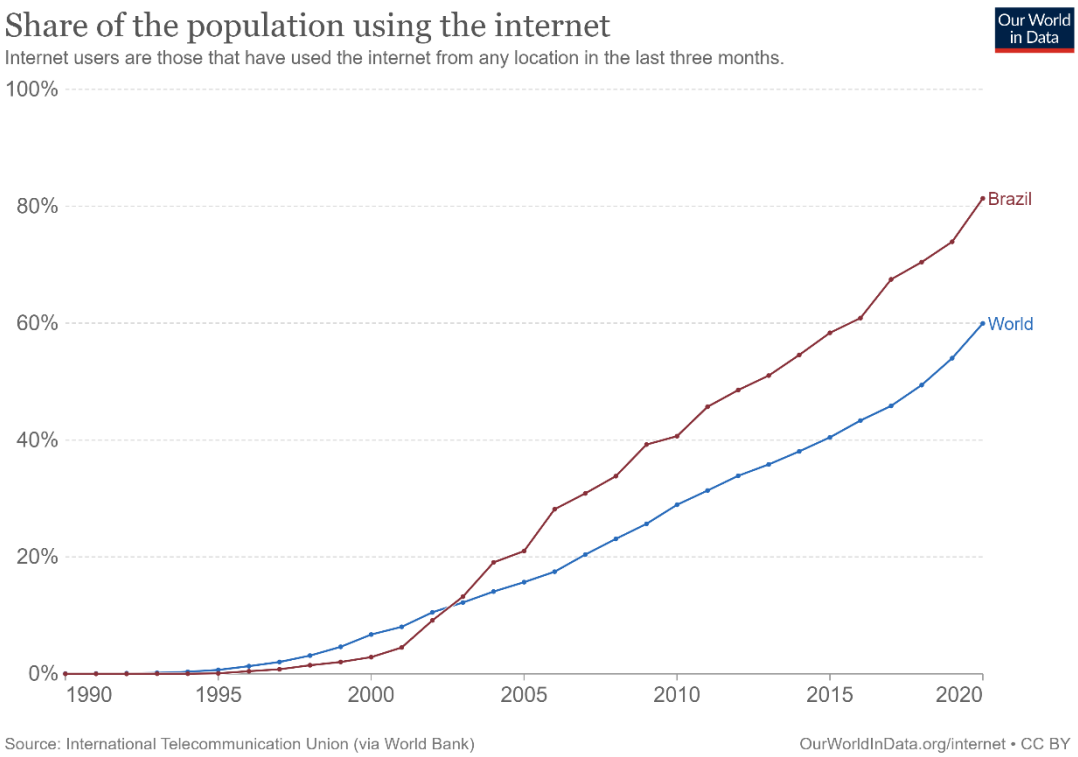
Figura 1. Usuários ativos por mês no Facebook



Fonte: Disponível em : < https://wearesocial.com/uk/blog/2022/01/digital-2022-another-year-of-bumper-growth-2/> Acesso em 09/01/2023.

A proporção da população mundial com acesso à Internet tem crescido constantemente nos últimos anos. De acordo com o relatório "State of the Internet" da Akamai Technologies, em dezembro de 2020, cerca de 57,9% da população mundial tinha acesso à Internet (ver figura 2), o que equivale a cerca de 4,5 bilhões de pessoas. De modo que 58,6% das pessoas com acesso à internet, são usuários ativos da rede social.

Figura 2. População mundial e brasileira com acesso à internet no período de 1990 a 2020



Fonte: Disponível em: <https://ourworldindata.org/internet#:~:text=Between%202014%20and%202016%20the,of%20the%20world%20are%20online> Acesso e: 09/01/2023.

Ainda de acordo com a ONU, até 2030 o restante da população que ainda não tem acesso à internet deve ser alcançada, e com isso o alcance das redes sociais deve continuar crescendo.

No Brasil esses dados são mais otimistas, ainda de acordo com os dados coletados pelo Our World In Data, cerca de 81,34% dos brasileiros possuíam acesso à internet até o ano de 2020, ou cerca de 173 milhões de brasileiros, e de acordo com a plataforma Statista, que mantém os registros de acessos das redes sociais por país, nesse mesmo período a rede social possuía cerca de 144 milhões de usuários ativos no país, ou cerca de 83,24% da população com acesso à internet.

Esses dados servem para mostrar a presença das redes sociais hoje na população, não apenas brasileira, e mostrar como as redes sociais têm afetado como as pessoas se relacionam umas com as outras em nível pessoal, mas como devido alguns fatores externos, e sobretudo a pandemia de COVID 19, que se iniciou no ano de 2020, fez crescer também a velocidade de digitalização de empresas no país.

Como o crescimento no número de usuários das redes sociais e de interações nesses meios, tem-se surgido um novo debate, o que devemos ou não permitir nas redes sociais, e por possuir o controle dos meios de comunicação, as companhias que detêm os direitos dessas redes (Google, Facebook, Twitter) e suas subsidiárias estão realizando o papel de moderadores dos discursos em suas plataformas (ROSA, 2015).

De certo modo, essa moderação acaba ocorrendo apenas com comentários que são contra as diretrizes dessas empresas, e buscam inibir comentários que podem ser considerados como racistas, discursos de ódio, possíveis *Fake News*, material ligado a pornografia (vale apontar que cada rede social possui uma tolerância diferente a respeito desse tipo de material) e pedofilia.

Esse controle é feito através de denúncias por usuários das redes em junção com um agente (inteligência artificial), responsável por analisar as denúncias e definir em qual categoria o comentário se encaixa, e se deve ser removido da plataforma ou o usuário deve receber algum tipo de notificação.

Entretanto, comentários fora do padrão detectado pela plataforma podem se enquadrar em comentários indesejados, ou discurso de ódio, e nesses casos as redes sociais permitem que os usuários filtrem os comentários manualmente, e tomem as ações desejadas, apagando ou respondendo o comentário quando necessário.

Segundo o DT Index 2020 (Índice de Transformação Digital da Dell Technologies 2020), aproximadamente 87,5% das empresas instaladas no país, iniciaram alguma realização voltada a transformação digital nesse ano, ficando acima da média mundial de 80%.

Com o crescente número de empresas migrando para os meios digitais, têm-se feito necessário um novo profissional, responsável pela manutenção da imagem das empresas nas plataformas de comunicação, que devem, em muitos casos, ler os comentários públicos, e listá-los em desejáveis e indesejáveis.

Esses são problemas que não afetam apenas as empresas nas redes sociais, mas têm afeto muitos empresários, atores e outras figuras públicas, que poderiam se beneficiar de uma ferramenta que ajudasse a realizar uma manutenção em suas redes sociais, podendo não apenas detectar esses comentários, como também apaga os comentários e denunciar o autor desses atos.

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo será mostrado as principais pesquisas que envolvem a extração de dados de redes sociais via API, e como esses dados foram tratados, e quais passos podem auxiliar no tratamento dos dados pelo software.

3.1 Python para análise de sentimentos

Python é uma linguagem de programação de alto nível e interpretada, o que significa que é fácil de ler e escrever. Ela foi criada por Guido van Rossum no início dos anos 1990 e tem sido amplamente utilizada em diversos campos, como ciência de dados, desenvolvimento de aplicativos web e automação de tarefas. Python é conhecida por ser uma linguagem de programação de fácil aprendizado e por ter uma comunidade ativa e amigável (BIRD; LOPER, 2002).

Devido sua ampla gama de bibliotecas e frameworks, é possível usar a linguagem para o desenvolvimento de aplicações webs, análise de dados e aprendizado de máquina. Esses fatores contribuíram para que Python fosse a linguagem de programação mais usada no mundo na data de dezembro de 2022, segundo o TIOBE Index e o PYPL Index.

3.1.1 NLTK (Natural Language Toolkit)

Python é uma linguagem de programação muito popular que pode ser usada para realizar várias tarefas, incluindo análise de sentimentos. A análise de sentimentos é o processo de examinar dados, geralmente texto, para determinar as emoções ou opiniões que estão sendo expressas. Isso pode ser útil em várias áreas, como marketing, opinião pública e até mesmo saúde mental (BIRD, 2006).

Para realizar análise de sentimentos com Python, você pode usar bibliotecas especializadas, como a NLTK (Natural Language Toolkit) ou a TextBlob. Essas bibliotecas fornecem ferramentas e recursos que podem facilitar o processo de análise de sentimentos. Além disso, existem vários tutoriais e exemplos de código disponíveis online que podem ajudá-lo a começar. É importante lembrar que a análise de sentimentos é uma área complexa da ciência de dados e pode exigir conhecimento em outras áreas, como processamento de linguagem natural e aprendizado de máquina.

A NLTK (Natural Language Toolkit) é uma biblioteca de Python que fornece ferramentas e recursos para trabalhar com dados de linguagem natural. Algumas das principais funções da NLTK incluem:

- Processamento de texto: a NLTK oferece ferramentas para dividir texto em tokens (palavras ou pontuação), remover pontuação, detectar stopwords (palavras comuns que não acrescentam significado ao texto), entre outras coisas.
- Análise de sentimentos: a NLTK inclui um classificador de sentimentos pré-treinado que pode ser usado para detectar emoções em texto. Também é possível treinar o classificador com seus próprios dados para melhorar sua precisão.
- Processamento de linguagem natural: a NLTK inclui ferramentas para realizar tarefas comuns de processamento de linguagem natural, como taggeamento de partes da fala e identificação de entidades.
- Aprendizado de máquina: a NLTK fornece alguns algoritmos de aprendizado de máquina básicos que podem ser usados para treinar modelos com seus próprios dados.

Essas são apenas algumas das principais funções da NLTK. A biblioteca é muito completa e inclui muitas outras ferramentas e recursos que podem ser úteis para trabalhar com dados de linguagem natural. Essa biblioteca também permite o uso de métodos de análise Naive Bayes, que é um algoritmo de aprendizado de máquina que é comumente usado em análise de sentimentos e outras tarefas de processamento de linguagem natural (ZHANG, p. 3 2004). Ele é chamado de "naive" porque assume que todas as características (palavras no caso de análise de sentimentos) são independentes entre si, o que nem sempre é verdade. No entanto, mesmo com essa simplificação, ele ainda pode fornecer resultados bastante bons em muitas situações.

Foi realizado um estudo para a detecção do discurso de ódio em postagens usando o Naive Bayes, e o para os dados coletados o classificador apresentou uma precisão de cerca de 73% na classificação dos dados.

3.2 Utilização de APIs para extração de dados em redes sociais.

APIs são siglas em inglês para "Application Programming Interfaces" ou, em português, "Interfaces de Programação de Aplicativos". Em resumo, são conjuntos de ferramentas e protocolos que permitem que diferentes aplicativos e sistemas se comuniquem e troquem dados de forma eficiente. Isso permite que os desenvolvedores criem funcionalidades e integrações em seus aplicativos, sem precisar recriar todo o código do zero. Exemplos

comuns de APIs incluem serviços de mapas, serviços de pagamento e plataformas de redes sociais.

As APIs podem ser úteis neste processo, pois permitem que os desenvolvedores de aplicativos acessem e façam uso de grandes conjuntos de dados de forma fácil e eficiente. Isso pode ser especialmente útil se a empresa ou organização que está realizando o datamining possui acesso a dados de diferentes fontes, como redes sociais, bases de dados comerciais ou outras fontes de dados públicas. Utilizando APIs, os desenvolvedores podem acessar esses dados de forma organizada e consistente, o que facilita o processo de datamining.

Desse modo foram realizados estudos com APIs para extração de dados de redes sociais, e até mesmo tentando enumerar as diversas aplicações disponíveis dentro das grandes plataformas de interação social, muitas buscando entender como o usuário interage com a plataforma.

Dentre as principais aplicações, vemos que a extração de dados, principalmente comentários, ocupam as pesquisas mais citadas dentre este ramo, mapear o comportamento do usuário, e mapear possíveis comportamentos nocivos a comunidade.

No trabalha Identificando a Toxicidade dentro dos comentários de Youtube, foi feito um estudo, mostrando a toxicidade nos comentários de canais com posicionamentos favoráveis e contra a OTAN, o que gerou um banco de dados de 1.424 vídeos pró-OTAN com 8.276 comentários e 3.461 vídeos contrários a OTAN, com 46.464 comentários (OBADIMU; MEAD; HUSSAIN; AGARWAL, p. 229, 2019)

Para uma classificação mais precisa da toxicidade dentro da plataforma, foi usada a ferramenta Perspective API, uma ferramenta criada para o fim de combater o comportamento tóxico na internet, desenvolvida pelo Projeto Jigsaw do google e o time Contra Abuso de Tecnologias.

Jigsaw é um projeto da Google que visa desenvolver tecnologias de inteligência artificial para resolver problemas complexos relacionados à segurança e violação de direitos humanos. O projeto foi criado em 2010 como uma iniciativa da Google Ideas, e desde então tem desenvolvido diversas ferramentas e tecnologias, como o sistema Perspective, que utiliza aprendizado de máquina para identificar comentários abusivos e degradantes em plataformas de mídia social. Jigsaw também tem trabalhado em projetos relacionados à proteção de jornalistas, prevenção de ataques cibernéticos e outros temas relacionados à segurança online.

Esse modelo usa rede neural convolucional, que é um tipo de rede neural artificial que foi especialmente desenvolvida para lidar com dados que têm uma estrutura espacial, como imagens. Elas se baseiam na ideia de "convolução", que é um processo matemático que

permite que a rede capture padrões complexos em dados de entrada. Isso é feito através de várias camadas de "neurônios" que são treinados para reconhecer padrões específicos em dados de entrada. Redes neurais convolucionais são amplamente utilizadas em aplicações de visão computacional, como reconhecimento de imagens e processamento de linguagem natural.

A análise dos dados mostrou que apenas trinta e cinco porcento (35%) dos comentários em vídeos favoráveis a OTAN mostraram comportamento não tóxico, enquanto nos vídeos contrários a OTAN, esse número foi ainda menor, chegando oitenta e sete porcento (87%). Desse modo a tecnologia usada para essa análise se mostrou efetiva em detectar comentários tóxicos na rede social

Existem diversos trabalhos que tratam a respeito do uso de APIs para extrair dados em conjunto com ferramentas de análise de sentimentos que buscam categorizar os dados obtidos em positivos e negativos.

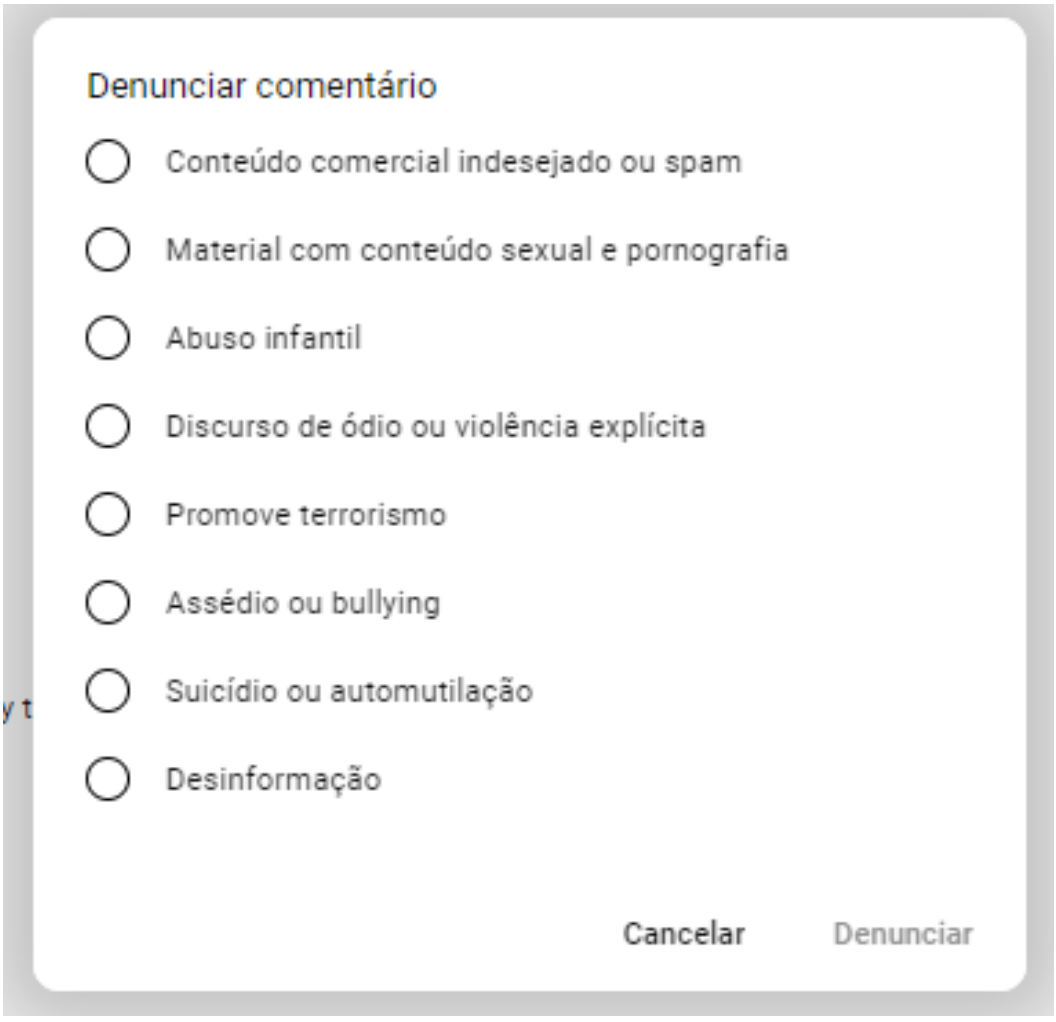
Mas não foi encontrado nenhum trabalho que busque, após a análise inicial dos dados, deletar de forma automática os comentários da plataforma, e criar um serviço de monitoramento responsável por interceptar novas mensagens e categorizá-las antes que possa haver um contato com o restante do público da plataforma.

4 PROPOSIÇÃO

A utilização de tecnologias de análise de sentimento é comum nas principais redes sociais, mas devido ao alto volume de serviço e aos altos custos computacionais envolvidos na análise de todos os comentários postados, essas plataformas ainda dependem dos usuários para classificar os comentários como uma medida inicial.

A denúncia de comentários em redes sociais é o processo de notificar a equipe responsável pelo site sobre comentários que violam as regras ou políticas da rede social. Isso é feito para garantir que a plataforma mantenha um ambiente seguro e agradável para todos os usuários. A denúncia pode ser feita através de um link ou opção de denúncia disponível na própria postagem ou comentário (ver figura 3), ou através de contato direto com a equipe de suporte da rede social. É importante lembrar que cada rede social tem suas próprias regras e políticas, então é sempre importante lê-las antes de fazer uma denúncia.

Figura 3 – Formulário de denúncia de comentário

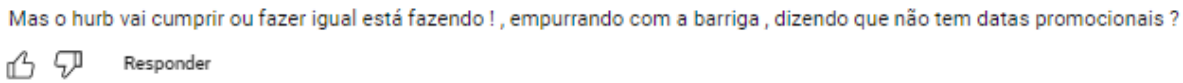


Fonte: Formulário retirado da plataforma Youtube em 09/01/2023

O conteúdo dos comentários denunciados pode ser usado como dados para treinar modelos de aprendizado de máquina para identificar e classificar comentários inapropriados de acordo com as regras da rede social. Isso pode ajudar a melhorar a precisão e eficácia dos sistemas de análise de sentimento e contribuir para manter um ambiente seguro e agradável nas redes sociais.

Outro ponto importante, é o de que muitas vezes, mesmo respeitando as regras de uso das plataformas, o comentário pode não se adequar ao que o produtor de conteúdo deseja, por isso a maioria das redes sociais oferece a opção de excluir comentários individuais em suas próprias postagens. Para fazer isso, basta encontrar a opção de exclusão na própria postagem ou comentário e selecioná-la. Na figura 04, vê-se um comentário que pode ser indesejado pelo canal que fez a postagem do conteúdo, e que não fere nenhum dos termos de conduta da plataforma, desse modo, para filtrar mensagens como essa, os produtores de conteúdo contam com uma equipe responsável por filtrar essas mensagens, respondendo ou as retirando da plataforma.

Figura 4 – Exemplo de comentário indesejado



Fonte: Comentário Retirado do vídeo Seu guia de viagem para Punta Cana I Hora do Break <https://www.youtube.com/watch?v=zcHWZBLqzBc&ab_channel=Hurb> 14/11 às 10:31

Pensando nesses casos, foi pensada uma ferramenta capaz de realizar a extração de todos os comentários da página, isso pode ser feito com a ajuda de APIs. Com os comentários extraídos, a equipe do produtor de conteúdo deve classificar os comentários extraídos em comentários positivos e negativos.

Com os comentários já classificados, será treinado um modelo de aprendizado de máquina, que possa identificar novos comentários como positivos e negativos, bem como tomar ações baseadas nessa classificação, como curtir o comentário ou deletá-lo.

E por fim, será desenvolvida uma aplicação que recebe as novas mensagens da rede social, para que, assim que um novo comentário seja postado na rede social, uma ação seja tomada

5 METODOLOGIA

Nessa seção trataremos os métodos utilizados para a realização do trabalho, passando pelas ferramentas utilizadas, e as etapas realizadas, mostrando como elas foram realizadas e quais os principais desafios de cada uma das etapas.

5.1 Aquisição de dados

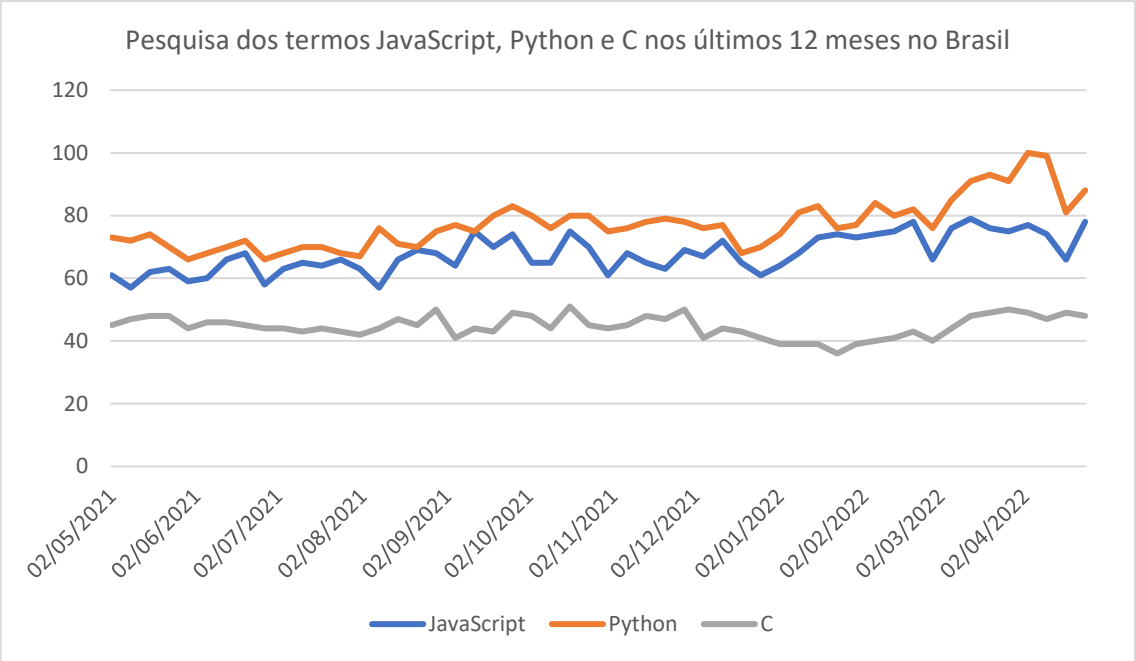
Para a aquisição dos dados foram necessárias duas principais escolhas, a primeira sendo qual a linguagem de programação estaríamos usando e qual rede social seria alvo da extração de dados via API.

5.1.1 Python

A linguagem de programação escolhida foi a linguagem Python, sendo uma linguagem de alto nível, possuindo sintaxe simples, fazendo com que a linguagem fosse de fácil aprendizagem, tornando a linguagem uma das mais populares, de acordo com a pesquisa da plataforma *stackoverflow*, uma das maiores plataformas que buscam ajudar programadores em suas dúvidas, cerca de 48,24% de seus usuários possuem experiência com Python, estando atrás apenas de Javascript (64,96%) e HTML / CSS (56,07%).

Mas se analisarmos os dados segundo o google trends (ver figura 5), que mostra as tendências de pesquisas para os termos selecionados vemos um cenário um pouco diferente.

Figura 5 – Linguagens mais pesquisadas no Google



Fonte – < https://trends.google.com.br/trends/explore?geo=BR&
q=%2Fm%2F05z1_,%2Fm%2F02p97,%2Fm%2F01t6b> Acesso em 02/04/2022.

Pode-se observar que a linguagem Python no Brasil fica à frente da demais linguagens, vale notar que as notas são relativas ao ponto de maior procura dos dados dentro do período analisado, e não em valores absolutos, considerando a primeira semana de abril com a semana de maior busca pela linguagem de programação, segundo a plataforma, a média de pesquisas do Python 13,2% maior que as pesquisas pela linguagem de programação Javascript, ao olharmos para o cenário global esses dados são maiores ainda, sendo que a média das buscas em Python é de 62,3% maior que as buscas por Javascript.

Devido a sua popularidade, Python é uma linguagem de programação muito versátil que pode ser usada para uma ampla variedade de propósitos. Algumas das principais aplicações da linguagem Python incluem:

- Desenvolvimento de aplicativos web, como sites e sistemas de gerenciamento de conteúdo (CMS).
- Análise de dados e ciência de dados, como trabalhar com grandes conjuntos de dados e criar modelos de machine learning.
- Desenvolvimento de aplicativos de desktop e dispositivos móveis.
- Automatização de tarefas, como envio de emails ou processamento de arquivos.

- Desenvolvimento de jogos.
- Criação de scripts para automatizar tarefas no sistema operacional.
- Desenvolvimento de sistemas de banco de dados.

Essas são apenas algumas das principais aplicações da linguagem Python. A linguagem é amplamente utilizada em muitas outras áreas devido à sua facilidade de uso e à sua ampla comunidade de desenvolvedores.

5.1.2 Bibliotecas

As principais bibliotecas usadas para a extração e tratamento de dados foram as bibliotecas, de Requests que é uma biblioteca que permite que os desenvolvedores façam solicitações HTTP de forma fácil e intuitiva. Com ela, é possível enviar solicitações GET, POST, PUT, DELETE e outras, bem como lidar com respostas HTTP, incluindo erros e códigos de status.

A biblioteca requests é útil em aplicações que precisam se comunicar com outros serviços da web, como APIs, por exemplo. Ela também pode ser usada para fazer o download de arquivos de uma URL ou enviar dados para um servidor.

Para instalar a biblioteca do Python, basta instala-lá com o comando *pip install requests.*

Essa biblioteca permite fazer chamadas HTTP, ao servidor do Meta, as chamadas seguem o padrão de documentação encontrada na página de desenvolvedores da empresa. Para obter a autorização da conta, deve-se cadastrar um conta de desenvolvedor nos sites, e para ter acesso a conta desejada, com permissões para editar e comentar, devemos também ter acesso a conta da plataforma e solicitar uma chave de acesso que será usada da requisição de novas informações.

Também seria necessário a biblioteca json, que é uma biblioteca que permite que os desenvolvedores trabalhem com dados no formato JSON (JavaScript Object Notation) de forma fácil e intuitiva. O formato JSON é um formato de dados que é comumente usados para troca de dados entre aplicações web, pois é fácil de ler e escrever tanto para humanos quanto para máquinas.

A biblioteca json é útil em aplicações que precisam trabalhar com dados em formato JSON, como APIs que retornam dados em formato JSON ou aplicações que precisam enviar dados em formato JSON para outros serviços.

Por fim a biblioteca pandas do Python, que é uma biblioteca que fornece estruturas de dados e ferramentas de análise de dados de alto desempenho e fáceis de usar. Ela é muito útil para trabalhar com grandes conjuntos de dados em Python e fazer análises avançadas, como agregação, filtragem e transformação de dados.

A biblioteca pandas é amplamente utilizada em aplicações de ciência de dados e análise de dados, como por exemplo para limpar e preparar dados para modelos de machine learning, ou para criar gráficos e visualizações a partir de dados.

5.1.3 Meta API

A Meta oferece uma API de dados chamada Graph API, que permite que desenvolvedores acessem dados do Facebook e Instagram de forma controlada e segura. A Graph API é amplamente utilizada em aplicações que precisam integrar-se com o Facebook, como por exemplos aplicativos de login com o Facebook, aplicativos que compartilham conteúdo no Facebook, ou aplicativos que analisam dados das redes sociais.

A Graph API oferece acesso a dados como publicações, comentários, usuários, páginas e eventos do Facebook, entre outros. Para usá-la, é preciso criar uma conta de desenvolvedor no Facebook e seguir os passos de autenticação e autorização para obter um token de acesso válido. Depois disso, você pode usar a API para fazer solicitações HTTP e obter os dados desejados.

É importante observar que o acesso à Graph API está sujeito a limitações e políticas de privacidade, portanto é recomendado ler e entender essas políticas antes de usar a API. Além disso, o acesso a dados do Facebook pode ser restrito a determinados usuários ou tipos de conteúdo, então é possível que nem todos os dados desejados estejam disponíveis.

Dentre os pontos desejáveis para essa aplicação, a API do Meta possui suporte para extração de diversos dados dentro da plataforma, incluindo os comentários em casa post, que serão necessários para a análise de sentimentos. A possibilidade de tomada de ações para os comentários analisados, como a exclusão do comentário, ou resposta no caso de comentários positivos, e a possibilidade de se comunicar com um servidos por meio de Webhooks, que são uma forma de permitir que aplicações se comuniquem de forma assíncrona usando solicitações HTTP. Eles permitem que uma aplicação avise outra aplicação sobre eventos ocorridos, sem precisar que a segunda aplicação faça periodicamente uma solicitação para verificar se houve mudanças. Isso torna o processo mais rápido e eficiente, pois a aplicação que recebe o aviso pode agir imediatamente em vez de esperar por uma nova solicitação.

Para usar webhooks, é preciso configurar a aplicação que envia os avisos (chamada de "aplicação do remetente") para enviar solicitações HTTP para a aplicação que recebe os avisos (chamada de "aplicação do destinatário") quando determinados eventos ocorrerem. A aplicação do destinatário deve então ter um ponto de entrada específico (geralmente uma URL) que possa receber e processar essas solicitações.

Webhooks são muito úteis em aplicações que precisam se comunicar em tempo real, como por exemplo em aplicações de mensagens, notificações e alertas. Eles também são amplamente utilizados em integrações com serviços da web, como APIs, que podem enviar avisos sobre mudanças em dados ou status de processamento.

5.1.4 Jupyter

Para usar as ferramentas listadas foi escolhido o aplicativo jupyter que é um aplicativo de código aberto que permite criar e compartilhar documentos que contêm código, equações, visualizações e texto. Ele é amplamente utilizado em ciência de dados, análise de dados e aprendizado de máquina, pois permite combinar código executável, resultados e explicações em um único documento, o que é útil para documentar e compartilhar o processo de análise.

O Jupyter é chamado assim porque suporta várias linguagens de programação, incluindo Julia, Python e R. Ele é baseado em um aplicativo chamado IPython, que é uma implementação de console interativo de Python. O Jupyter é executado em um navegador da web e pode ser usado tanto localmente quanto em servidores remotos.

Existem várias vantagens em usar o Jupyter para ciência de dados, análise de dados e aprendizado de máquina:

- Interatividade: o Jupyter permite que você execute código em células individuais de um documento, o que é útil para testar pequenos trechos de código e visualizar resultados rapidamente.
- Documentação: o Jupyter permite que você combine código, resultados e explicações em um único documento, o que facilita a documentação do processo de análise e torna mais fácil compartilhar o código e os resultados com outras pessoas.
- Suporte para várias linguagens: o Jupyter suporta várias linguagens de programação, incluindo Python, R, Julia e muitas outras, o que permite escolher a linguagem mais adequada para a tarefa em questão.

- Ferramentas de visualização: o Jupyter vem com várias bibliotecas de visualização, como Matplotlib e Seaborn, que permitem criar gráficos e outras visualizações de dados de maneira fácil e rápida.
- Colaboração: o Jupyter pode ser usado em conjunto com outras ferramentas de colaboração, como o GitHub, o que facilita o trabalho em equipe e o compartilhamento de código e resultados

5.2 Tratamento de dados

Para o tratamento de dados foi usada a biblioteca NLTK (natural language tool kit), é uma base de dados para treinamento, que será usada para medir a acurácia dos dados analisados e escolher o melhor classificador a ser usado.

Para usar o NLTK (Natural Language Toolkit), primeiro é necessário instalá-lo em seu ambiente Python. Isso pode ser feito usando o gerenciador de pacotes pip com o seguinte comando:

$$pip\ install\ nltk$$

Uma vez instalado, você pode importar o NLTK em seu código Python usando a seguinte sintaxe.

$$import\ nltk$$

Pode-se iniciar o tratamento de dados com um processo conhecido com tokenização, que é o processo de dividir um texto em unidades menores chamadas tokens. Essas unidades podem ser palavras, pontuação, símbolos ou outras sequências de caracteres que tenham significado para a tarefa em questão. A tokenização é uma etapa importante em muitas tarefas de processamento de linguagem natural, pois permite que o texto seja processado e analisado de forma mais precisa e eficiente.

O NLTK (Natural Language Toolkit) é uma biblioteca de processamento de linguagem natural em Python que fornece uma função chamada word_tokenize para realizar a tokenização de um texto. Essa função tokeniza o texto em palavras, usando regras de tokenização padrão que são comuns em muitas línguas. Por exemplo, ao tokenizar o texto "Este é um exemplo de tokenização", a função word_tokenize retornará a seguinte lista de tokens:

$$['Este', 'é', 'um', 'exemplo', 'de', 'tokenização']$$

Com a tokenização realizada a segunda etapa é realizar a retirada das stop words, Uma stop word é uma palavra comum que é frequentemente usada no idioma, mas não contribui

com muito significado em uma frase ou sentença. Essas palavras são frequentemente ignoradas em tarefas de processamento de linguagem natural, como análise de sentimentos e extração de informações, pois podem prejudicar a precisão dos resultados. O NLTK (Natural Language Toolkit) fornece uma lista pré-definida de stop words em inglês, e outros idiomas que podem ser usadas para remover essas palavras de um texto.

$$filtered\_tokens = [token\ for\ token\ in\ tokens\ if\ token\ not\ in\ stop\_words]$$

Outro mecanismo muito usado é o steming das palavras, O stem de palavras no NLTK é a forma reduzida de uma palavra que é obtida removendo sufixos e prefixos. Isso é útil em aplicações de processamento de linguagem natural, como classificação de documentos e agrupamento de palavras semelhantes, pois ajuda a simplificar a representação de uma palavra e a identificar sua raiz.

Por exemplo, a palavra "running" pode ser reduzida ao seu stem "run", enquanto a palavra "organize" pode ser reduzida ao seu stem "organ". Isso permite que palavras nadiferentes, mas com a mesma raiz, sejam tratadas de maneira semelhante pelo modelo de processamento de linguagem natural.

Agora, deve-se calcular o número de vezes que cada palavra é citada dentro da base de dados, isso será usado futuramente para prever a possibilidade de uma palavra pertencer ao grupo de mensagens negativas ou positivas. Também é necessário encontrar o número de palavras únicas presente nos dados coletados, isso pode ser feito com o método set(), que é um lista de elementos únicos no Python.

Com esses dados, é possível implementar o algoritmo de Naive Bayes em uma base de dados para a conferência dos resultados.

Essa base de dados pode ser encontrada de duas maneiras, ou criando uma base de dados própria, analisando os dados dentro da plataforma, e os categorizando em positivos e negativos, de acordo com o ambiente de coleta. Ou buscando algumas bases de dados próprias para testes na internet, há uma versão que disponibiliza dados da de mais de 142,8 milhões de reviews de produtos na Amazon com notas fornecidas pelos clientes.

A base de dados utilizada deve conter o padrão de uma coluna referente ao texto a ser analisado, uma segunda coluna referente a nota (ou emoção) atribuída pelo usuário ao comentário, e por fim, uma coluna referente a avaliação do algoritmo, que terá que se aproximar do valor atribuído pelo usuário, e ao alcançar um nível de acerto elevado, deverá ser levado ao ambiente de produção, no qual receberá mais dados e aumentará sua precisão.

5.3 Naive Bayes

O método de Naive Bayes é considerado um método mais simples de classificação de texto, porém muito eficaz, com ele separamos os dados previamente classificados em positivos e negativos, e mapeamos quais palavras aparecem em cada uma das mensagens, e a frequência das palavras em cada divisão. Com essa divisão feita, consideramos a chance de uma palavra do grupo das palavras positivas aparecer em uma mensagem positiva.

$$P(palavra|mensagem\ positiva) = \frac{frequência\ da\ palavra\ nas\ mensagens\ positivas}{total\ de\ palavras\ analisadas\ das\ mensagens\ positivas}$$

Desse modo, têm-se um mapeamento para todas as palavras apresentadas nas mensagens positivas, agora deve-se repetir o processo com as palavras nas mensagens de origem negativas.

$$P(palavra|mensagem\ negativa) = \frac{frequência\ da\ palavra\ nas\ mensagens\ negativas}{total\ de\ palavras\ analisa\ das\ mensagens\ negativas}$$

Também é necessário calcular a chance de uma mensagem qualquer, participar da lista mensagens positivas e negativas, para isso consideramos o número de mensagens positivas e o número total de mensagens.

$$P(positiva) = \frac{total\ de\ mensagens\ positivas}{total\ de\ mensagens}$$

E para a probabilidade de uma mensagem não pertencer as mensagens positivas.

$$P(negativo) = 1 - P(positiva)$$

De modo que para cada nova mensagem analisada, será considerada a probabilidade de pertencimento a um dos grupos como, sendo que o que possuir a maior probabilidade será considerada considerado como uma mensagem desse grupo.

$$P(mensagens\ positivas) = P(positivas) * \prod P(palavras|positivas)$$

$$P(mensagens\ negativas) = P(negativas) * \prod P(palavras|negativas)$$

Mas como pode ser visto nas equações do Naive Bayes, pode ser considerado ingênuo, por não considerar a ordem nas quais as palavras são colocadas, de modo que mesmo sendo eficiente para casos mais simples, teve-se sempre ter em mente as limitações dessa análise (CASTRO, p. 15, 2019).

5.4 Webhook

Para o monitoramento de novos comentários foi criado um webhook, que é um mecanismo de notificação que permite que um aplicativo ou serviço envie uma mensagem para outro aplicativo ou serviço sempre que algum evento ocorrer. Ele é usado para permitir que aplicativos e serviços "se comuniquem" uns com os outros e permitem que eles se integrem de maneira mais fácil e rápida.

Por exemplo, imagine que você tem um aplicativo de gerenciamento de tarefas e deseja ser notificado sempre que uma tarefa for concluída. Você pode configurar um webhook para enviar uma mensagem para o seu aplicativo sempre que uma tarefa for concluída. Dessa forma, você não precisa ficar constantemente verificando se as tarefas foram concluídas; em vez disso, você só precisa aguardar a notificação do webhook.

Webhooks são geralmente usados para integrações entre aplicativos e serviços, permitindo que eles se comuniquem uns com os outros de maneira mais fácil e rápida. Eles são muito úteis para automatizar tarefas e permitir que os aplicativos e serviços reajam a eventos em tempo real.

Para criar um webhook com o Meta API, será necessário:

- Definir o endereço do webhook e o tipo de evento que deseja monitorar. Você pode definir o endereço do webhook em qualquer URL pública que esteja acessível pela Internet. O evento que se deseja monitorar pode ser qualquer tipo de evento que o Web API suporte, como uma alteração em um registro ou um novo usuário sendo criado.
- Criar uma função de retorno de chamada para lidar com o evento monitorado. Esta função de retorno de chamada deve ser capaz de processar os dados do evento e, em seguida, tomar a ação adequada. Por exemplo, pode-se usar a função de retorno de chamada para enviar uma notificação por email ou atualizar um banco de dados.
- Registrar o webhook com o Web API. Para fazer isso, é preciso enviar uma solicitação HTTP POST para o Web API, passando o endereço do webhook e o tipo de evento que deseja monitorar. O Web API retornará um ID de inscrição que você poderá usar para gerenciar ou excluir o webhook no futuro.
- Testar o webhook. Depois de ter registrado o webhook com sucesso, você pode testá-lo disparando o evento monitorado manualmente. Isso permitirá

que você veja se a função de retorno de chamada está sendo executada corretamente e se está tomando a ação adequada.

Para atualizações do tipo comentário, será enviada uma mensagem com a seguinte estrutura (ver figura 6).

Figura 6 – Mensagem enviada com novos comentários na plataforma

```
{
  "field": "comments",
  "value": {
    "from": {
      "id": "232323232",
      "username": "test"
    },
    "media": {
      "id": "123123123",
      "media_product_type": "FEED"
    },
    "id": "17865799348089039",
    "parent_id": "1231231234",
    "text": "This is an example."
  }
}
```

Fonte - < https://developers.facebook.com/apps/372800017088065/webhooks/> Acesso em: 09/01/2023

Com os dados recebidos via o webhook, basta analisar o campo de *text* enviado pela ferramenta, e caso seja o comentário seja marcado como um comentário negativo, é preciso enviar uma solicitação HTTP DELETE para a URL do comentário que desejado. A URL do comentário será algo como https://api.example.com/comments/{id}, onde {id} é o ID do comentário que que será excluído.

Verifique a resposta da solicitação. A Meta API retornará um código de resposta HTTP 204 (No Content) se o comentário for excluído com sucesso. Se ocorrer algum erro, a Meta API retornará um código de resposta HTTP adequado e uma mensagem de erro explicando o problema.

6 RESULTADOS E DISCUSSÕES

6.1 Aquisição de dados

A aquisição de dados é feito junto à API do Meta, para isso deve-se primeiro criar um conta de desenvolvedor na plataforma, essa conta será vincula automaticamente a conta na rede social, possuindo acesso as páginas ligadas à conta.

Para iniciar o processo, deve-se criar um aplicativo, na plataforma, esse método é padrão para a aquisição de dados das principais plataformas de interação social (TikTok, Youtube, Twitter), deve-se criar o aplicativo conforme as informações desejadas, e informações do usuário e conta a ser analisada.

Com o aplicativo configurado, devemos acessar na aba de ferramentas, explorador de Graph API, e, no item usuário ou página, selecionar a opção ‘token do aplicativo’, desse modo o token de acesso apresentado na página será criado, note que esse token possui duração de 24 horas, e deverá ser alterado por um token mais permanente.

6.1.1 Chamadas da API

Para as chamas da API, deve-se usar a biblioteca *requests*, que segundo a documentação oficial, permite ao usuário enviar chamadas HTTP de modo fácil usando o método JSON.

Para iniciar uma chamada, segundo a documentação do Meta, deve-se usar o protocolo HTTPs, e o domínio, graph.facebook.com, seguido da versão sobre a qual o aplicativo foi construído,

6.1.2 Extração dos comentários.

Segundo a documentação do Meta, os dados de comentários devem ser extraídos uma publicação por vez, desse modo, seria necessário os dados de ID dos posts desejados e token de acesso válido para realizar a requisição. Para o Obtenção da ID das publicações do IG, deve-se conhecer primeiro, o ID da conta desejada no Instagram, e enviar uma requisição https, com essa informação, e será retornada um JSON, com os campos de link dos posts,

legenda escrita pelo usuário na postagem da mídia, tipo da mídia, data da postagem e ID dos posts.

Com os dados de ID dos posts de uma conta do Instagram, agora é possível, a requisição dos comentários é feita de modo individual para cada post, de modo que o modelo da requisição com a biblioteca *requests,* com o método get, seguindo o modelo.

$$https://graph.facebook.com/\{versão\ do\ aplicativo\}/\{id\ do\ post\}/comments/$$

Ao passar pelos dados de todos os posts coletados, o dado foi exportado pelo formato .csv(comma separeted value) para análises futuras.

6.2 Análise de dados

Antes de seguir com a classificação dos dados coletados, será necessário treinar um algoritmo para que seja capaz de classificar as mensagens analisadas, isso pode ser feito usando dados próprios, e nesse caso seria necessário realizar, além da coleta, uma classificação manual dos dados coletas, esse método permitiria uma maior precisão nas análises, e ainda permitiriam que as análises fossem feitas respeitando as especificidades de encontradas em diferentes contas. Como também usando dados classificados por terceiros. Para esse trabalho foi usada uma base de dados referentes a comentários nos produtos da Amazon, a base de dados escolhida possui 10.000 dados classificados com notas de 1 até 5, sendo 1 considerado produtos com péssimas avaliações, e 5 com as melhores avaliações. Dentre essa coletânea de foram separas dos dados com notas maior ou igual a três, considerando essas revisões positivas, e notas menor que três como revisões negativas, a base de dados inicial possui certa de 84,7% de dados positivos, e 15,3% de dados com avaliações negativas (ARAUJO; GONÇALVES, ; BENEVENUTO, 2013).

Para usar os dados de teste, primeiro deve ser separado valores para treinar o modelo desejado, no caso foram feitos testes usando uma base auxiliar com 5%, 10% e 20% dos dados da base de revisão para conferir se haveria melhorias significativas com o aumento do número de dados na base de teste.

Tabela 1 – Proporção dos dados coletados

Avaliações	Base de dados	Treino (5%)	Treino (10%)	Treino (20%)
Positivas	84,8%	83,0%	84,4%	85,0%
Negativas	15,2%	17,0%	15,6%	15,0%

Fonte: Autoria própria

Tendo realizado os métodos de tratamento de dados anteriormente discutidos nesse trabalho, o classificador foi é treinado com a ajuda da função apply_features e nltk.NaiveBayesClassifier.train.

O método nltk.apply.features(), essa é uma função auxiliar que pode ser usada para preparar os dados de treinamento ou teste para o uso em um classificador do NLTK. Ela recebe como argumento uma função de "extração de características" (também conhecida como "função de extrator de atributos") e uma lista de observações, e retorna uma lista de tuplas, onde cada tupla representa uma observação e contém dois elementos: um dicionário de atributos (também chamado de características) e a classe a qual essa observação pertence.

A função de extração de características é uma função que recebe uma observação como entrada e retorna um dicionário de atributos para essa observação. Por exemplo, a função de extração de características pode extrair as palavras mais frequentes de um texto e usá-las como atributos.

A função apply_features é útil porque ela permite que os dados sejam preparados de forma rápida e eficiente, ao mesmo tempo em que aplica a função de extração de características a todas as observações.

A função train do NLTK é um método de treinamento para classificadores Naive Bayes. Ele é usado para criar um classificador Naive Bayes a partir de uma lista de tuplas, onde cada tupla representa uma observação e contém dois elementos: um dicionário de atributos (também chamado de características) e a classe a qual essa observação pertence.

Com a base treinada, a função accuracy do NLTK pode ser usada para avaliar a precisão de um classificador. Ela recebe como argumentos um classificador e uma lista de tuplas, onde cada tupla representa uma observação e contém dois elementos: um dicionário de atributos (também chamado de características) e a classe a qual essa observação pertence. A função accuracy retorna um número de 0 a 1, indicando a porcentagem de observações que foram classificadas corretamente pelo classificado

Para os testes feitos usando as bases de dados de treinamento, formadas por 5%, 10% e 20% do total de dados da base de dados escolhida, foram encontradas as seguintes acurácias.

Tabela 2 – Precisão do classificador conforme base de treino

Base de Treino	5%	10%	20%
Precisão	84,1%	84,2%	85%

Fonte: Autoria própria

7 CONCLUSÃO

Neste capítulo, serão apresentadas as conclusões e comentários obtidos a partir do desenvolvimento do trabalho de graduação, incluindo uma avaliação dos objetivos e propostas estabelecidos. Também será apresentada uma aplicação do trabalho no contexto do curso de Engenharia de Controle e Automação.

Com o aumento do número de pessoas que usam as redes sociais, também tem se feito cada vez mais necessário que haja um cuidado maior sobre os comentários que são feitos em cada perfil, e como isso pode afetar seus usuários. Mas devido ao alto tráfego de pessoas nas principais plataformas de comunicação, isso têm se mostrada uma tarefa bastante desafiadora.

Desse modo, muitos perfis acabam fazendo um trabalho manual de leitura, classificação e exclusão dos comentários em suas plataformas. E isso pode ser necessário especificamente em perfis de grande expressão na internet.

Esse trabalho foi realizado com o intuito inicial de criar uma ferramenta que fosse capaz de auxiliar o combate a comentários indesejados nas redes sociais, pelo uso de técnicas de análise de sentimentos, ajudando assim nos pontos levantados anteriormente.

Para isso foi escolhido o classificador naive bayes para criar uma primeira versão da ferramenta que fosse capaz de analisar os dados, e categorizar em positivos e negativos. Também foi possível criar um serviço, que se comunique com a Meta API, empresa responsável pelas redes sociais do facebook e instagram, monitore por novos comentários, e tome uma ação baseada nesses comentários.

Nesse sentido o sistema proposto foi capaz de alcançar os resultados desejados, mas vale notar que existem classificadores melhores que o de Naive Bayes, que foi adotado sobretudo, por falta de um equipamento com maior capacidade computacional, e buscando reduzir os custos de análises que poderiam surgir com um alto número de requisições.

REFERÊNCIAS BIBLIOGRÁFICAS

ARAUJO, M.; GONÇALVES, P.; BENEVENUTO, F.; **Métodos para Análise de Sentimentos no Twitter.,** 2013, Disponível em: https://homepages.dcc.ufmg.br/~fabricio/download/webmedia13.pdf , . Acesso em: 16/02/2026.

BIRD, S.; **NLTK: The Natural Language Toolkit**, Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, 2002

BIRD, S.; LOPER, E.; **NLTK: The Natural Language Toolkit**, Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, p.69–72, 2006

CASTRO, L.; **Um Estudo Empírico Sobre Técnicas Para Detecção De Discursos De Ódio Em Postagens Públicas Escritas Em Português,** Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, 2019

OBADIMU, A.; MEAD, E.; HUSSAIN, M. N.; AGARWAL, N.; **Identifying Toxicity Within YouTube Video Comment,** 12th International Conference, SBP-BRiMS, p.229-238, 2019

RISH, I.; **An empirical study of the naive Bayes classifier,** IJCAI 2001 Work Empir Methods Artif Intell, 2001

ROSA, R. L.; **Análise de Sentimentos e afetividade de textos extraídos das redes sociais**. Tese (Doutorado em Sistemas Digitais) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2015.

ZHANG, H.; **The Optimality of Naive Bayes,** Faculty of Computer Science - University of New Brunswick, 2004

ANEXO A – Autenticação na API

```
import requests
import json

def getCreds():
  creds = dict()
  creds['access_token'] = 'SECRERT'
  creds['client_id'] = 'ID'
  creds['client_secret'] = 'CLIENT_SECRET'
  creds['graph_domain'] = 'https://graph.facebook.com'
  creds['graph_version'] = 'v12.0'
  creds['endpoint_base'] = creds['graph_domain'] + '/'  +
 creds['graph_version'] + '/'  # base endpoint with domain and version
  creds['debug'] = 'no'  # debug mode for api call
  creds['page_id'] = '101065334962117'  # users page id
  creds['instagram_account_id'] = 'ACCOUNT_ID'  # users instagram account id
  creds['ig_username'] = 'ACCOUNT_NAME'  # ig username

  return creds


def makeApiCall(url, endpointParams, debug='no'):
  """ Request data from endpoint with params

  Args:
      url: string of the url endpoint to make request from
      endpointParams: dictionary keyed by the names of the url parameters
  Returns:
      object: data from the endpoint
  """

  data = requests.get(url, endpointParams)  # make get request

  response = dict()  # hold response info
  response['url'] = url  # url we are hitting
  response['endpoint_params'] = endpointParams  # parameters for the endpoint
```

```
  response['endpoint_params_pretty'] = json.dumps(endpointParams, indent=4)  #
pretty print for cli
  response['json_data'] = json.loads(data.content)  # response data from the
api
  response['json_data_pretty'] = json.dumps(response['json_data'], indent=4)
 # pretty print for cli

  if ('yes' == debug):  # display out response info
    displayApiCallData(response)  # display response

  return response  # get and return content


def displayApiCallData(response):
  """ Print out to cli response from api call """

  print ("\nURL: ")  # title
  print (response['url'])  # display url hit
  print ("\nEndpoint Params: ")  # title
  print (response['endpoint_params_pretty'])  # display params passed to the
endpoint
  print ("\nResponse: ")  # title
  print (response['json_data_pretty'])  # make look pretty for cli
```

ANEXO B – Requisição de comentários

```
#!/usr/bin/env python
# coding: utf-8

# In[8]:


from defines import getCreds, makeApiCall
import pandas as pd

from datetime import datetime ,timedelta
from getstory import getInstaStory
import requests

def getUserMedia(params):



    endpointParams = dict()  # parameter to send to the endpoint
    endpointParams['fields'] =
'id,caption,media_type,media_url,permalink,thumbnail_url,timestamp,username'
 # fields to get back
    endpointParams['access_token'] = params['access_token']  # access token

    url = params['endpoint_base'] + params['instagram_account_id'] + '/media'
 # endpoint url

    return makeApiCall(url, endpointParams, params['debug'])  # make the api
call

def getMediaComments (params , pagingUrl=''):

    endpointParams['access_token'] = params['access_token']
    url = params['endpoint_base'] + params['post_id'] + '/comments'
    if ('' == pagingUrl):  # get first page
        url = params['endpoint_base'] + params['post_id'] + '/comments'  #
endpoint url
    else:  # get specific page
```

```
        url = pagingUrl  # endpoint url

    return makeApiCall(url, endpointParams, params['debug'])

def getMediaInsights(params):

    endpointParams = dict()  # parameter to send to the endpoint
    endpointParams['metric'] = params['metric']  # fields to get back
    endpointParams['access_token'] = params['access_token']  # access token

    url = params['endpoint_base'] + params['post_id'] + '/insights'  #
endpoint url

    return makeApiCall(url, endpointParams, params['debug'])  # make the api
call


def getUserInsights(params):


    endpointParams = dict()  # parameter to send to the endpoint
    #endpointParams['metric'] =
'follower_count,impressions,profile_views,reach,website_clicks,text_message_cl
icks'  # fields to get back
    #endpointParams['period'] = 'day'  # period
    #endpointParams['since'] = '1630454400'
    #endpointParams['until'] = '1632268800'
    #endpointParams['metric'] = 'audience_city , audience_country'  # fields
to get back
    #endpointParams['metric'] = 'audience_city ,
audience_country,audience_gender_age,audience_locale,online_followers'  #
fields to get back
    endpointParams['metric'] = params['metric']
    endpointParams['period'] = params['period']
    if params['metric'] ==
'audience_city,audience_country,audience_gender_age,audience_locale':
        pass
    else:
```

```
        endpointParams['since'] = params['since']
        endpointParams['until'] = params['until']
    endpointParams['access_token'] = params['access_token']  # access token

    url = params['endpoint_base'] + params['instagram_account_id'] +
'/insights'  # endpoint url

    return makeApiCall(url, endpointParams, params['debug'])  # make the api
call




# In[9]:




# In[2]:


i = 0

feed_id = []
engajamento = []
impressoes = []
alcance = []
salvos = []
imagens = [engajamento , impressoes , alcance , salvos]


video_id = []
engajamento_video = []
impressoes_video = []
alcance_video = []
salvos_video = []
visualizacoes = []
```

```
videos = [engajamento_video , impressoes_video , alcance_video , salvos_video
, visualizacoes]


params = getCreds()  # get creds
response = getUserMedia(params)  # get users media from the api


# In[10]:


insta = getInstaPosts ()


# In[12]:









# In[ ]:


feed_id = []
engajamento = []
impressoes = []
alcance = []
salvos = []
likes = []
imagens = [engajamento , impressoes , alcance , salvos]


video_id = []
engajamento_video = []
impressoes_video = []
alcance_video = []
salvos_video = []
visualizacoes = []
likes_video = []
```

```
videos = [engajamento_video , impressoes_video , alcance_video , salvos_video
, visualizacoes]

for i , post_id in enumerate (insta['ID do Post']):
    params['post_id'] = post_id

    if 'VIDEO' == insta ['Tipo de Midia'][i]:
        params['metric'] = 'engagement,impressions,reach,saved,video_views'

    else:

        params['metric'] = 'engagement,impressions,reach,saved'
    try:
        insights = getMediaInsights(params)
        for j , insight in enumerate(insights['json_data']['data']) :
            if 'VIDEO' == insta ['Tipo de Midia'][i]:
                videos[j].append(int(insight['values'][0]['value']))
            else:
                imagens[j].append(int(insight['values'][0]['value']))

        if 'VIDEO' == insta ['Tipo de Midia'][i]:
            video_id.append(post_id)
        else:
            feed_id.append(post_id)
    except:
        pass


# In[ ]:


engajamento = imagens[0].copy()
impressoes = imagens[1].copy()
alcance = imagens[2].copy()
salvos = imagens[3].copy()

engajamento_video = videos[0].copy()
impressoes_video = videos[1].copy()
alcance_video = videos[2].copy()
```

```
salvos_video = videos[3].copy()
visualizacoes = videos[4].copy()


# In[ ]:


insights_fotos = pd.DataFrame()
insights_videos = pd.DataFrame()

insights_fotos['ID do post'] = feed_id
insights_fotos['Engajamento'] = engajamento
insights_fotos['Impressões'] = impressoes
insights_fotos['Alcance'] = alcance
insights_fotos['Salvos'] = salvos

insights_videos['ID do post'] = video_id
insights_videos['Engajamento'] = engajamento_video
insights_videos['Impressões'] = impressoes_video
insights_videos['Alcance'] = alcance_video
insights_videos['Salvos'] = salvos_video
insights_videos['Visualizações'] = visualizacoes


# In[ ]:


insights_fotos.to_excel('insights_imagens.xlsx' , index= False)


# In[ ]:


insights_videos.to_excel('insights_videos.xlsx' , index= False)


# In[ ]:



ate = []
```

```
desde = []
ate.append (datetime.now())
desde.append (datetime.now() - timedelta(29))

for i in range (24):
    ate.append(desde[i])
    desde.append(desde[i]- timedelta(29))

ate_timestamp = [int(datetime.timestamp(data)) for data in ate]
desde_timestamp = [int(datetime.timestamp(data)) for data in desde]



# In[ ]:


periodo = ['day','days_28','week','lifetime']
metrics_lifetime_periodless =
'audience_city,audience_country,audience_gender_age,audience_locale'
metrics_lifetime_withperiod = 'online_followers'
metrics_day =
'email_contacts,get_directions_clicks,impressions,profile_views,reach,text_mes
sage_clicks,website_clicks'

metrics_days_week = 'impressions,reach'


# In[ ]:


agora = datetime.now()
futuro = agora + timedelta(15)
passado = agora - timedelta(15)
futuro = int(datetime.timestamp(futuro))
passado = int(datetime.timestamp(passado))

params['period'] = 'day'
params['metric'] = 'follower_count'
params['since'] = desde_timestamp[0]
```

```
params['until'] = ate_timestamp[0]

metricas_mensal = [[],[]]
insight = getUserInsights (params)
for i , data in enumerate(insight['json_data']['data']):
    for value in data['values']:
        metricas_mensal[0].append(value['value'])
        metricas_mensal[1].append(value['end_time'])

validade_mensal = pd.DataFrame()
validade_mensal['Ganho de seguidores'] = metricas_mensal [0]
validade_mensal['Data'] = metricas_mensal [1]
validade_mensal.to_excel('ganho_de_seguidores.xlsx' , index = False)


# In[ ]:


validade_mensal = pd.DataFrame()
validade_mensal['Ganho de seguidores'] = metricas_mensal [0]
validade_mensal['Data'] = metricas_mensal [1]

validade_mensal.to_excel('ganho_de_seguidores.xlsx' , index = False)


# In[ ]:


metricas = [[],[],[],[],[],[],[],[]]
email_contacts = []
get_directions_clicks = []
impressions = []
profile_views = []
reach = []
text_message_clicks = []
website_clicks = []
data = []

params['period'] = 'day'
```

```
params['metric'] = metrics_day

for i in range (len (desde_timestamp)):
    a = len(desde_timestamp) - i - 1
    params['since'] = desde_timestamp[a]
    params['until'] = ate_timestamp[a]
    insights = getUserInsights(params)
    try:
        for j , insight in enumerate(insights['json_data']['data']) :
            for value in insight['values']:
                metricas[j].append(int(value['value']))
                if j == (1):
                    data.append(value['end_time'])
    except:
        pass


# In[ ]:


email_contacts = metricas[0].copy()
get_directions_clicks = metricas[1].copy()
impressions = metricas[2].copy()
profile_views = metricas[3].copy()
reach = metricas[4].copy()
text_message_clicks = metricas[5].copy()
website_clicks = metricas[6].copy()


# In[ ]:


metricas_diarias = pd.DataFrame()
metricas_diarias ['Contatos de Email'] = email_contacts
metricas_diarias ['Clicks em Como Chegar'] = get_directions_clicks
metricas_diarias ['Impressões'] = impressions
metricas_diarias ['Visualizações'] = profile_views
metricas_diarias ['Alcance'] = reach
metricas_diarias ['Clicks em mensagem de texto'] = text_message_clicks
```

```
metricas_diarias ['Clicks em website'] = website_clicks
metricas_diarias ['Data'] = data
metricas_diarias.to_excel ('metricas_diarias.xlsx' , index= False)


# In[ ]:


metricas = [[],[],[],[]]
cidade = [[],[]]
pais = [[],[]]
sexo_idade = [[],[]]
localidade = [[],[]]

params['period'] = 'lifetime'
params['metric'] = metrics_lifetime_periodless

insights = getUserInsights(params)
try:
    for j , insight in enumerate(insights['json_data']['data']) :
        metricas[j].append(list(insight['values'][0]['value'].items()))

except:
    pass


# In[ ]:


cidade = [[],[]]
pais = [[],[]]
sexo_idade = [[],[]]
localidade = [[],[]]

for i in range (len (metricas[0][0])):
    cidade[0].append(metricas[0][0][i][0])
    cidade[1].append(metricas[0][0][i][1])
    pais[0].append(metricas[1][0][i][0])
    pais[1].append(metricas[1][0][i][1])
```

```
    try:
        localidade[0].append(metricas[3][0][i][0])
        localidade[1].append(metricas[3][0][i][1])
    except:
        pass
    try:
        sexo_idade[0].append(metricas[2][0][i][0])
        sexo_idade[1].append(metricas[2][0][i][1])
    except:
        pass

cidade_do_seguidor = pd.DataFrame()
cidade_do_seguidor['Cidade'] = cidade[0]
cidade_do_seguidor['Seguidores'] = cidade[1]
pais_do_seguidor = pd.DataFrame()
pais_do_seguidor['Pais'] = pais[0]
pais_do_seguidor['Seguidores'] = pais[1]
sexo_do_seguidor = pd.DataFrame()
sexo_do_seguidor['Sexo e Idade'] = sexo_idade[0]
sexo_do_seguidor['Seguidores'] = sexo_idade[1]
localidade_do_seguidor = pd.DataFrame()
localidade_do_seguidor['Sexo e Idade'] = localidade[0]
localidade_do_seguidor['Seguidores'] = localidade[1]

cidade_do_seguidor.to_excel('cidade.xlsx' , sheet_name='Cidades' ,
index=False)
pais_do_seguidor.to_excel('pais.xlsx' , sheet_name='Pais' , index=False)
sexo_do_seguidor.to_excel('sexo_idade.xlsx' , sheet_name='Sexo e Idade' ,
index=False)
localidade_do_seguidor.to_excel('localidade.xlsx' , sheet_name='Localidade' ,
index=False)


# In[ ]:


params['period'] = 'lifetime'
params['metric'] = metrics_lifetime_withperiod
params['since'] = desde_timestamp[0]
```

```
params['until'] = int(datetime.timestamp(datetime.now() - timedelta(2)))
online_followers = []
seguidores = []
end_time = []
insight = getUserInsights (params)
for i , data in enumerate(insight['json_data']['data'][0]['values']):
    teste = list(data['value'].items())
    seguidores = []
    for valor in teste:
        seguidores.append(valor[1])
    online_followers.append(seguidores)
    end_time.append(data['end_time'])




# In[ ]:


seguidores_online = pd.DataFrame()
for i , data in enumerate (end_time):
    seguidores_online['{}'.format(str(data))] = online_followers[i]
seguidores_online.to_excel('seguidores_online.xlsx' , index= False)


# In[ ]:


stories = getInstaStory()
story_data = [[],[],[],[],[],[]]
story_id = []

params['metric'] = 'exits,impressions,reach,replies,taps_forward,taps_back'
for i , post_id in enumerate (stories['ID do Post']):
    params['post_id'] = post_id
    insights = getMediaInsights(params)
    try:
        for j , insight in enumerate(insights['json_data']['data']) :
                story_data[j].append(int(insight['values'][0]['value']))
```

```
        story_id.append(post_id)
    except:
        pass
exits = []
impressions = []
reach = []
replies = []
taps_f = []
taps_b = []

exits = story_data[0]
impressions = story_data[1]
reach = story_data[2]
replies = story_data[3]
taps_f = story_data[4]
taps_b = story_data[5]

stories_df = pd.DataFrame()
stories_df['Story ID'] = story_id
stories_df['Saídas'] = exits
stories_df['Impressões'] = impressions
stories_df['Alcance'] = reach
stories_df['Respostas'] = replies
stories_df['Cliques avançar'] = taps_f
stories_df['Cliques voltar'] = taps_b

stories_df.to_excel('stories_insights.xlsx' , index = False)


# In[ ]:


post = []
comment_id = []
comment = []
timestamp = []

for post_id in insta['ID do Post']:
    params['post_id'] = post_id
```

```
    comments = getMediaComments (params)
    while (True):
        for comentario in comments['json_data']['data']:
            comment_id.append (comentario['id'])
            comment.append (comentario['text'])
            timestamp.append (comentario['timestamp'])
            post.append(post_id)
        try:
            comments = getMediaComments(params,
comments['json_data']['paging']['next'])
        except:
            break




# In[ ]:


post = []
comment_id = []
comment = []
timestamp = []


params['post_id'] = insta['ID do Post'][1]
comments = getMediaComments (params)
while (True):
    for comentario in comments['json_data']['data']:

        comment_id.append (comentario['id'])
        comment.append (comentario['text'])
        timestamp.append (comentario['timestamp'])
    try:
        comments = getMediaComments(params,
comments['json_data']['paging']['next'])
    except:
        break
```

```
# In[ ]:


comentarios = pd.DataFrame()
comentarios['Id do post'] = post
comentarios['Id do comentário'] = comment_id
comentarios['Comentário'] = comment
comentarios['Publicação'] = timestamp
comentarios.to_excel ('comentarios_insta.xlsx', index = False)


# In[ ]:
```

ANEXO C – Análise dos dados

```
#!/usr/bin/env python
# coding: utf-8

# In[171]:


import pandas as pd
import numpy as np
import nltk
from nltk.tokenize import word_tokenize

import tensorflow as keras
from tensorflow.keras.preprocessing.text import Tokenizer

reviews = pd.read_csv("reviews.csv")
reviews.dropna(inplace=True)
reviews.drop(columns=['ProductId', 'UserId', 'ProfileName','Summary' ,'Time',
'HelpfulnessNumerator', 'HelpfulnessDenominator'], inplace=True)


# In[172]:



reviews.columns = ['Sentimento', 'Frase']
reviews.replace([4, 5],'Good', inplace=True)
reviews.replace([3],'Good', inplace=True)
reviews.replace([1, 2],'Bad', inplace=True)
training_data = reviews.sample(n=(int(len(reviews)/10)))
training_data.reset_index(inplace=True, drop=True)


# In[173]:


print((training_data.Sentimento.value_counts() / training_data.shape[0]) *
100)
```

```
# In[174]:


print(len(reviews)/10)


# In[175]:


print((reviews.Sentimento.value_counts() / reviews.shape[0]) * 100)


# In[176]:


from nltk.tokenize import word_tokenize

def tokenizarFrase(dataframe):
    frases = []

    for i, frase in enumerate(dataframe["Frase"]):
        tokens = word_tokenize(frase)
        frases.append((tokens,dataframe["Sentimento"][i]))
    return frases
reviews_tokenized = tokenizarFrase(reviews)
training_data_tokenized = tokenizarFrase(training_data)


# In[177]:


from nltk.corpus import stopwords

def removeStopWord(data):
    filtered_data = []
    stop_words = stopwords.words('english')
    stop_words.append(".")
    stop_words.append(",")
    stop_words.append("/")
```

```
    stop_words.append("<")
    stop_words.append(">")
    stop_words.append("!")
    stop_words.append("?")
    for frase, sentimento in data:
        filtered_tokens = [token for token in frase if token not in
stop_words]
        filtered_data.append((filtered_tokens, sentimento))
    return filtered_data


reviewsNoStopWords = removeStopWord(reviews_tokenized)
trainingNoStopWords = removeStopWord(training_data_tokenized)


# In[178]:


from nltk.stem import PorterStemmer

def wordStemmer(data):
    filtered_data = []
    stemmer = PorterStemmer()
    for frase, sentimento in data:
        new_frase = []
        for word in frase:
            new_frase.append(stemmer.stem(word))
        filtered_data.append((new_frase, sentimento))
    return filtered_data

reviewsStemmed = wordStemmer(reviewsNoStopWords)
trainingStemmed = wordStemmer(trainingNoStopWords)


# In[179]:


def busca_Palavras(frases):
    todas_Palavras = []
```

```
    for (palavras, sentimento) in frases:
        todas_Palavras.extend(palavras)
    return todas_Palavras


# In[180]:


palavras_treinamento = busca_Palavras(trainingStemmed)
palavras_teste = busca_Palavras(reviewsStemmed)


# In[181]:


def busca_frequencia(palavras):
    palavras = nltk.FreqDist(palavras)
    return palavras

frequencia_treinamento = busca_frequencia(palavras_treinamento)
frequencia_teste = busca_frequencia(palavras_teste)


# In[ ]:




# In[ ]:




# In[ ]:
```

```
# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:




# In[182]:


def busca_palavras_unicas(frequencia):
    freq = frequencia.keys()
    return freq

palavras_unicas_treinamento = busca_palavras_unicas(frequencia_treinamento)
palavras_unicas_teste = busca_palavras_unicas(frequencia_teste)
```

```
# In[183]:


frequencia_teste.most_common(10)


# In[184]:


def extrator_palavras(documento):
    # Utilizado set() para associar a variavel doc com o parâmetro que esta
chegando
    doc = set(documento)
    caracteristicas = {}
    for palavras in palavras_unicas_treinamento:
        caracteristicas['%s' % palavras] = (palavras in doc)
    return caracteristicas


# In[185]:


def extrator_palavras_teste(documento):
    doc = set(documento)
    caracteristicas = {}
    for palavras in palavras_unicas_teste:
        caracteristicas['%s' % palavras] = (palavras in doc)
    return caracteristicas


# In[186]:


base_completa_treinamento = nltk.classify.apply_features(extrator_palavras,
trainingStemmed)

base_completa_teste = nltk.classify.apply_features(extrator_palavras_teste,
reviewsStemmed)


# In[187]:
```

```
classificador = nltk.NaiveBayesClassifier.train(base_completa_treinamento)

# In[191]:

base_completa_treinamento[0]

# In[188]:


print(classificador.show_most_informative_features(10))

# In[189]:

print(nltk.classify.accuracy(classificador, base_completa_teste))

# In[ ]:
```