INSTITUTO FEDERAL GOIANO - CAMPUS MORRINHOS
CURSO BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL QUINTINO FONSECA GUIMARÃES

ANÁLISE DE SENTIMENTO E PREVISIBILIDADE DE SUCESSO DE NOTÍCIAS PUBLICADAS NO SITE DO IF GOIANO

MORRINHOS - GO
2025

RAFAEL QUINTINO FONSECA GUIMARÃES

ANÁLISE DE SENTIMENTO E PREVISIBILIDADE DE SUCESSO DE NOTÍCIAS PUBLICADAS NO SITE DO IF GOIANO

Monografia apresentada ao Curso Bacharelado em Ciência da Computação do Instituto Federal Goiano - Campus Morrinhos, como requisito parcial para obtenção de título de Bacharel em Ciência da Computação.
Área de concentração: Análise e Ciência de Dados.
Orientador: Prof. Dr. Jesmmer da Silveira Alves.

MORRINHOS - GO

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema Integrado de Bibliotecas do IF Goiano - SIBi

Guimarães, Rafael Quintino Fonseca
G963t Análise de Sentimento e Previsibilidade de Sucesso de Notícias Publicadas no Site do IF Goiano / Rafael Quintino Fonseca Guimarães. Morrinhos 2025.

131f. il.

Orientador: Prof. Dr. Jesmmer da Silveira.
Tcc (Bacharel) - Instituto Federal Goiano, curso de 0420196 - Bacharelado em Ciência da Computação - Morrinhos (especial) (Campus Morrinhos).
1. Análise de Notícias. 2. Análise de Sentimento. 3. Mineração de Texto. 4. Transformers. 5. Web Scraping. I. Título.

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Rafael Quintino Fonseca Guimarães

Matrícula:
2020104201940101

Título do trabalho:
Análise de Sentimento e Previsibilidade de Sucesso de Notícias Publicadas no Site do IF Goiano

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 18 / 01 / 2026

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

gov.br Documento assinado digitalmente
RAFAEL QUINTINO FONSECA GUIMARAES
Data: 20/12/2025 17:50:24-0300
Verifique em https://validar.iti.gov.br

Morrinhos - Goiás
Local

20 / 12 / 2025
Data

_____
Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

_____
Assinatura do(a) orientador(a)

gov.br Documento assinado digitalmente
JESMMER DA SILVEIRA ALVES
Data: 22/12/2025 07:49:52-0300
Verifique em https://validar.iti.gov.br

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 110/2025 - CCEG-MO/CEG-MO/DE-MO/CMPMHOS/IFGOIANO

ATA DE DEFESA DE TRABALHO DE CURSO (TC)

Aos dezoito dias do mês de dezembro de 2025, por meio de videoconferência, realizou-se a sessão pública de defesa de Trabalho de Curso do Curso Bacharelado em Ciência da Computação, do acadêmico Rafael Quintino Fonseca Guimarães, sob orientação do professor Jesmmer da Silveira Alves, intitulada ANÁLISE DE SENTIMENTO E PREVISIBILIDADE DE SUCESSO DE NOTÍCIAS PUBLICADAS NO SITE DO IF GOIANO. Compuseram a Banca Examinadora os professores:

Orientador

Professor Dr. Jesmmer da Silveira Alves

Membro 2

Professora Dra. Leila Roling Scariot da Silva

Membro 3

Jornalista Ma. Rejane da Silva Alves

Após a exposição oral, o candidato foi arguido pelos membros da banca, os quais reuniram-se reservadamente, e decidiram, **APROVADO**. Para constar, redigi a presente Ata, que aprovada por todos os presentes, vai assinada por mim, Orientador do TC, e pelos demais membros da banca.

_____

Prof. Orientador

Assinado eletronicamente

_____

Prof.(a)/Membro 2

Assinado eletronicamente

_____

Prof.(a)/Membro 3

Assinado eletronicamente

Documento assinado eletronicamente por:

- **Jesmmer da Silveira Alves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 20/12/2025 17:42:42.
- **Rejane da Silva Alves**, **JORNALISTA**, em 20/12/2025 17:55:31.
- **Leila Roling Scariot da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 20/12/2025 18:05:24.

Este documento foi emitido pelo SUAP em 20/12/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 777000
Código de Autenticação: 8c9d5cd50b

INSTITUTO FEDERAL GOIANO
Campus Morrinhos
Rodovia BR-153, Km 633, Zona Rural, SN, Zona Rural, MORRINHOS / GO, CEP 75650-000
(64) 3413-7900

DEDICATÓRIA

Dedico este trabalho aos meus pais, que sempre me apoiaram e incentivaram meus estudos.

AGRADECIMENTOS

Agradeço a Deus por me conceder a capacidade e a sabedoria necessárias para a elaboração deste trabalho. Sem Ele, nada poderia fazer (João 15:5).

Ao meu pai, Afonso, e à minha mãe, Fabiana, pelo apoio constante e incondicional dedicado aos meus estudos.

Expresso minha sincera gratidão ao Professor Dr. Jesmmer pela orientação competente na produção deste trabalho e pela constante dedicação em me apoiar no aperfeiçoamento da escrita.

RESUMO

Este trabalho analisa as notícias publicadas no site do Instituto Federal Goiano para identificar quais são as mais acessadas e as suas razões de sucesso na plataforma. Utilizou-se de técnicas de *web scraping* com a linguagem de programação Python e a biblioteca de automação de navegador, Selenium, para desenvolver um robô de software capaz de coletar os dados das notícias do site, e da plataforma Apache Airflow, para automatizar a sua execução, que foi feita diariamente. As notícias foram classificadas em categorias como editais e processos seletivos, analisando o título e a descrição das publicações com o *chatbot* ChatGPT, da série de LLMs, GPT da OpenAI. A avaliação da quantidade de acessos foi realizada por meio da linguagem Python e da biblioteca para análise de dados, Pandas, o que forneceu *insights* valiosos para a melhor compreensão dos padrões de interesse e preferências dos usuários. As técnicas de análise de dados que envolvem a escrita e a leitura de arquivos CSV, seleção e filtragem por classes de objetos e valores numéricos, manipulação de colunas de dados tabulados e ausentes, cálculo de estatísticas descritivas e Z-Score e de visualização de dados através de gráficos, foram utilizadas para extrair as informações mais importantes das publicações. Os modelos de inteligência artifical pré-treinados para tarefas de NLP, *Bidirectional Encoder Representations from Transformers* (BERT, em português, ”Representações de Codificador Bidirecional de Transformers”) da Google e *Bidirectional and Auto-Regressive Transformer* (BART, em português, ”Transformer Bidirecional e Auto-Regressivo”) da Meta, foram utilizados para realizar uma análise de sentimento das notícias, visando descobrir qual é a influência de textos com sentimento mais positivo na quantidade de acessos, e produzir uma nova descrição de cada notícia que seja mais bem avaliada pela análise de sentimento do modelo BERT, respectivamente. Os resultados da pesquisa indicam que os usuários do site demonstram maior interesse por notícias relacionadas a editais e processos seletivos. Ademais, verificou-se que publicações contendo uma única imagem de capa tendem a apresentar maior média de acessos. Por fim, identificou-se que os dias mais propícios para publicação de notícias no site do IF Goiano correspondem, aproximadamente, aos primeiros dias úteis da semana.

Palavras-chave: Análise de Notícias. Análise de Sentimento. Mineração de Texto. *Transformers*. *Web Scraping.*

ABSTRACT

This study analyzes the news published on the website of the Instituto Federal Goiano in order to identify which articles receive the most views and the reasons behind their success on the platform. Web scraping techniques were employed using the Python programming language and the Selenium browser automation library to develop a software robot capable of collecting news data from the website. The Apache Airflow platform was used to automate its execution, which was performed daily. The news articles were classified into categories such as public notices (editais) and selection processes, by analyzing the titles and descriptions of the publications with the ChatGPT chatbot, from the GPT series of LLMs by OpenAI. The evaluation of the number of accesses was conducted using Python and the data analysis library Pandas, which provided valuable insights for understanding user interest patterns and preferences. Data analysis techniques involving reading and writing CSV files, selecting and filtering by object classes and numerical values, handling tabulated and missing data, calculating descriptive statistics and Z-Scores, and visualizing data through charts were applied to extract the most relevant information from the publications. Pretrained artificial intelligence models for NLP tasks, such as Google's Bidirectional Encoder Representations from Transformers (BERT) and Meta's Bidirectional and Auto-Regressive Transformer (BART), were used to perform sentiment analysis on the news. The goal was to determine the influence of more positive language on the number of views and to generate a new description for each article that would receive higher sentiment ratings from the BERT model. The findings indicate that users of the website show greater interest in news related to public notices and selection processes. Additionally, it was observed that publications containing a single cover image tend to achieve a higher average number of views. Finally, the study identified that the most suitable days for publishing news on the IF Goiano website correspond, approximately, to the first business days of the week.

Keywords: News Analysis. Sentiment Analysis. Text Mining. Transformers. Web Scraping.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Este trabalho teve como objetivo analisar a quantidade de acessos das notícias publicadas no site institucional do IF Goiano e correlacionar com as suas principais características. Desta forma, buscou-se encontrar quais fatores são os mais importantes para que uma publicação atinja uma maior quantidade de acessos. A análise foi feita com as notícias encontradas no site entre 26 de Janeiro e 13 de Fevereiro de 2023 (19 dias).

Neste período, o total de publicações encontradas no site variou de 2.559, no primeiro dia, até 2.613 notícias, no último dia, o que resultou em uma média de quase 3 (três) novas publicações por dia. Esta média relativamente alta, é justificada por se tratar dos primeiros meses do ano, período em que ocorre o início do ano letivo e o retorno dos discentes às aulas.

Esta investigação visa ser útil para as campanhas de divulgação de notícias do instituto, fornecendo metodologias para fomentar o aumento da popularidade do conteúdo midiático publicado. A análise dos resultados poderá revelar qual é o padrão de comportamento dos usuários do site, a partir de uma verificação dos elementos textuais e visuais das notícias, que podem fazer com que a sua leitura se torne mais atrativa.

Conforme explicam Barbosa, Araújo e Aragão (2016, p. 634), o uso de imagens e conteúdo visual em conjunto com texto verbal, pode aderir mais sentido à mensagem que se deseja expressar. Nas palavras dos autores, “o letramento visual está diretamente relacionado ao entendimento da informação visual não apenas como um adorno da informação verbal, mas como um elemento semiótico que agrega sentido ao texto”.

Para exemplificar, pode-se mencionar a existência de imagens chamativas no cabeçalho ou no corpo da notícia. Geralmente, a presença de interação visual pode atrair mais a atenção do leitor, persuadindo-o a acessar uma determinada notícia e ler o seu conteúdo. O público jovem é corriqueiramente mais atraído por publicações que apresentam tais elementos, como posts, banners, gifs, vídeos curtos, etc.

Deste modo, as notícias que apresentam tais características se tornam mais propícias a atingirem um maior número de pessoas e, a partir do uso das estratégias certas, um público específico, como os jovens. Em virtude disso, buscou-se explorar as características das notícias e levantar hipóteses acerca da quantidade de acessos das

publicações, como o dia da semana e o período do dia da publicação, o assunto principal da notícia, a análise de sentimento do texto da notícia e a presença de imagens.

Esta exploração foi feita no Capítulo 6, através do uso de métodos e técnicas de análise e inferência de dados e das disciplinas da Ciência da Computação, como Processamento de Linguagem Natural, com enfoque na implementação de algoritmos na linguagem de programação Python, amplamente utilizada nesta área e com uma imensa comunidade ativa atualmente.

Os resultados da pesquisa são, finalmente, apresentados no Capítulo 7, em formato de gráficos e visualizações de dados, com o intuito de indicar o quantitativo de acessos das notícias, a data da coleta e da publicação (i.e., mês, ano, etc) por cada assunto identificado, bem como os valores obtidos pelo cálculo da análise de sentimento, fazendo um cruzamento das informações e apresentando uma descoberta acerca dos padrões de formação do texto e da composição das notícias.

1.1 MOTIVAÇÃO

O Instituto Federal Goiano divulga em seu site institucional e nas suas redes sociais (Instagram e Facebook), as notícias dos eventos que acontecem nos campi da instituição. Estes são os principais meios de compartilhamento de informações do instituto, compartilhamento este, que ocorre de forma gratuita na rede mundial, permitindo qualquer usuário na web, acessar, extrair e analisar o conteúdo das notícias de forma manual ou automatizada (por meio de ferramentas de *software*), de uma forma bastante simples.

As ferramentas de *software* utilizadas na automação de processos são compostas por bibliotecas, isto é, coleções de funções e recursos que promovem dependências organizadas, compartilhamento e reuso de código, disponíveis gratuitamente para as mais diversas linguagens de programação modernas, como o Python. Elas oferecem uma *Application Programming Interface* (ou API, em português, “Interface de Programação de Aplicação”), que disponibiliza código reutilizável para as mais diversas tarefas,

como automação de processos, análise de dados, geração de gráficos e *plots*, entre muitas outras. Vale salientar, que essas APIs, geralmente, possuem documentação extensiva sobre como utilizá-las, o que agiliza bastante o processo de desenvolvimento.

Desta forma, aliando a disponibilização livre dos dados das notícias nas redes sociais e no site do IF Goiano, com a existência de um vasto número de bibliotecas para coleta e análise de dados, despertou-se o interesse para a composição deste trabalho, uma vez que os seus resultados poderiam ser bastante promissores.

A pesquisa procurou identificar quais características colaboram para que uma notícia tenha muitos acessos no site. As características analisadas foram: o dia da semana da publicação, o momento da publicação (manhã, tarde ou noite), a presença ou ausência de imagem no cabeçalho e/ou no corpo da notícia, o assunto geral (e.g., editais e processos seletivos) e, por fim, a análise de sentimento da publicação.

1.2 CONTEXTUALIZAÇÃO

Uma técnica de coleta de dados que vem sendo bastante difundida nas áreas de análise, ciência e engenharia de dados é o *web scraping*. Utilizado para realizar a extração de matéria informacional bruta da rede mundial, é definido em poucas palavras, mas na essência de seu significado por Khder (2021) como “o procedimento de extração automática de dados de websites utilizando *software*”.

Este termo significa algo como “raspagem da web” na língua inglesa, e indica a criação de *scripts*[1] para automatizar o acesso e a coleta de dados de páginas da web, como por exemplo, a Wikipédia e o site do IF Goiano. Para aplicar essa técnica ao projeto, utilizou-se de uma das bibliotecas da linguagem de programação Python, de maior popularidade na comunidade da linguagem: o *Selenium*.[2]

Além de *web scraping*, a automatização da execução dos robôs de *software*

[1] Conjuntos de comandos em linguagem de programação interpretada, utilizados para automatizar tarefas ou realizar processamentos simples e rápidos.

[2] Esta biblioteca oferece um conjunto de métodos e objetos úteis para uma ampla gama de funções de controle de navegador, como a abertura automática de URLs e a filtragem de componentes HTML.

com a plataforma *Airflow*, permitiu acelerar bastante o processo de extração dos dados. Esta plataforma, permite realizar o agendamento e o monitoramento da execução de um script Python, que define um *workflow*[3] em formato de uma *Directed Acyclic Graph* (*DAG*, em português, “Grafo Acíclico Direcionado”).

Uma DAG é um tipo de objeto disponibilizado pela API do *Airflow*, que permite indicar quais serão as tarefas a serem executadas e quais são as dependências entre elas (i.e., sequenciais ou paralelas), de forma similar a um grafo. As tarefas, por sua vez, são definidas em termos de *operators* (em português, “operadores”), outro tipo de objeto da *API*, que define o tipo de código, por exemplo, *PythonOperator*, para código Python puro, *BashOperator*, para comandos de terminal, e assim por diante.

1.3 ESTRUTURA DO TRABALHO

O trabalho foi estruturado para oferecer uma visão geral do problema e, gradualmente, aprofundar-se em cada etapa do processo. O objetivo é conduzir o leitor até a abordagem final adotada para alcançar os resultados obtidos, que, por ora, atendem aos objetivos previamente estabelecidos, conforme o roteiro desenvolvido.

Apresenta-se, na sequência, a definição dos objetivos gerais e específicos do estudo, uma revisão da literatura relacionada ao tema, seguida da descrição da metodologia aplicada e da implementação dos algoritmos e códigos utilizados para a coleta e análise dos dados. Por fim, os resultados são discutidos quanto à sua aplicabilidade e coerência com os objetivos definidos.

[3]No contexto da Ciência de Dados, *workflows* correspondem a sequências de etapas seguidas pelo cientista de dados para processar conjuntos de dados e gerar resultados para análise ou inferência.

2 OBJETIVOS

Os objetivos deste trabalho conduziram a investigação científica, orientando tanto o desenvolvimento teórico, com a definição da metodologia e dos fundamentos iniciais, quanto a aplicação prática, que envolveu a construção dos algoritmos e sua implementação em linguagem Python.

2.1 OBJETIVO GERAL

Identificar os fatores que influenciam o volume de acessos às notícias publicadas no site do IF Goiano, analisando características estruturais, visuais, temáticas e linguísticas presentes nos textos.

2.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos apresentados a seguir delimitam as etapas analíticas adotadas para investigar, de forma sistemática, os fatores que influenciam o volume de acessos às notícias do IF Goiano, bem como para avaliar o comportamento e o desempenho de modelos de processamento de linguagem natural na análise e síntese desses conteúdos.

- Analisar se o dia da semana e o período do dia da publicação influenciam a quantidade de acessos às notícias do site do IF Goiano;
- Verificar se a presença de imagens (imagem de perfil e/ou imagens no corpo do texto) está associada a um maior número de acessos;
- Realizar a análise de sentimento dos títulos e das descrições das notícias utilizando o modelo BERT, a fim de avaliar se textos com polaridade mais positiva apresentam maior quantidade de acessos;
- Comparar a análise de sentimento do texto resumido por meio do modelo BART com a descrição produzida pela equipe de jornalismo, verificando se o texto gerado pela IA apresenta diferença significativa de polaridade em relação ao texto humano.

3 REVISÃO DA LITERATURA

No que diz respeito ao referencial teórico produzido por pesquisadores brasileiros, observa-se uma escassez de material relevante e de qualidade sobre o estudo da motivação por trás da quantidade de acessos alcançada por notícias publicadas nas principais redes sociais contemporâneas, como X (antigo Twitter), Facebook e Instagram, em especial quando se trata de publicações em portais institucionais, como o do IF Goiano. Tal lacuna dificulta a compreensão dos fatores que contribuem para o sucesso ou insucesso dessas notícias no ambiente digital.

A consolidação de informações padronizadas é essencial para comparar resultados de trabalhos científicos, com vistas a progredir na construção de um conhecimento sólido sobre o tema. Embora ainda existam poucos trabalhos exatamente alinhados ao escopo desta pesquisa, existem trabalhos correlatos (i.e. estudo de notícias) que podem facilmente contribuir para o avanço da pesquisa e a obtenção de resultados concretos.

Para contextualizar o tema, o estudo realizado por Alves (2023), investigou a fundo como se dá o processo de surgimento de um tipo de notícia bastante comum nos dias atuais, principalmente com o advento das IAs generativas, que são encontradas em quase toda rede social pública: as *fake news*. Ao analisar o texto de notícias coletadas entre 2013 e 2021, o autor pôde definir quais são as características mais recorrentes encontradas neste tipo de notícia e construir um dicionário para identificá-las.

Desta forma, é possível dizer se uma notícia é falsa, analisando o seu conteúdo a partir do dicionário de características principais das *fake news*. Esse método só é valido para as notícias coletadas naquele período e com assunto similar ao das notícias inseridas na base de dados. As notícias analisadas neste trabalho, diferentemente daquelas exploradas por Alves (2023), são de caráter estritamente verídicas e publicadas em nome do Instituto Federal Goiano, o que traz ao usuário, uma presunção de maior confiabilidade, um ponto relevante a se considerar na análise da quantidade de acessos.

Dentro do escopo das notícias que envolvem temas políticos e/ou financeiros, Costa (2023) investigou o sentimento de mais de 17.000 notícias coletadas entre 2020 e 2023 no site do G1, através do emprego de técnicas de Aprendizado de Máquina (*Machine Learning*) para entender a relação entre o tom emocional das notícias e o comportamento de investidores na Bolsa de Valores brasileira (Ibovespa).

Nessa linha de pensamento, o desempenho do mercado financeiro em questão foi analisado sob a perspectiva do teor das notícias que vigoraram no período considerado, procurando identificar altas e baixas nos preços conforme o sentimento do texto das notícias. Essa técnica também foi empregada neste trabalho para análise das notícias do site do IF Goiano, tendo um objetivo similar: inferir se as notícias mais positivas atraíram mais acessos ou aquelas mais negativas despertaram uma maior atenção do público.

A análise de notícias só é viável mediante a posse de seus dados estruturados, como títulos, subtítulos, corpo do texto, rodapés, data e hora de publicação (*timestamp*), além de métricas como número de acessos ou interações. A coleta manual de tais informações, dada sua escala e variabilidade, torna-se inviável. Por isso, a utilização de algoritmos implementados em linguagens de programação mostra-se essencial para automatizar essa tarefa. De forma geral, o processo de extração automatizada de dados da web, por meio da simulação de interações entre usuário, navegador e interface de dados, é amplamente conhecido como *web scraping*, termo que pode ser traduzido, de forma aproximada, como “raspagem da web”.

É importante destacar que, atualmente (em 2025), a maioria das plataformas e sites que disponibilizam notícias, não oferece APIs públicas para integração com aplicações externas. As APIs representam a forma mais eficiente e estruturada de coletar dados de outros sistemas, pois fornecem pontos de acesso (*endpoints*) pré-definidos e documentados que permitem realizar requisições via protocolo HTTP/HTTPS e obter os dados formatados, prontos para serem processados ou armazenados em bancos de dados.

Para contornar essa limitação, Mohr *et al.* (2023) utilizou a biblioteca Selenium da linguagem Python, para implementar algoritmos capazes de realizar *web scraping* em páginas que continham os dados necessários à análise proposta pelos autores. No estudo, a coleta dos dados foi automatizada por meio dessas três bibliotecas, abrangendo as seguintes plataformas:

- Sistema IBGE de Recuperação Automática (SIDRA)[4];
- Conab[5];
- IBGE Censo Demográfico de 2010[6].

Assim, observa-se a flexibilidade e a adaptabilidade dessas bibliotecas, que não se restringem a tipos específicos de plataformas ou fontes, podendo ser programadas para acessar uma ampla variedade de dados. Essas fontes incluem desde bancos de dados e arquivos estruturados, como CSV e Excel, até conteúdos não estruturados extraídos de websites, como ocorre nesta pesquisa.

De acordo com Alghamdi e Alfalqi (2015, p. 1), os avanços recentes em aprendizado de máquina e estatística possibilitaram o desenvolvimento de novas técnicas para identificar padrões de palavras em coleções de documentos e textos, fundamentadas em modelos probabilísticos hierárquicos. Entre essas técnicas, além das abordagens mais comuns em Processamento de Linguagem Natural, se destaca a modelagem de tópicos.

Os autores destacam ainda que a técnica de modelagem de tópicos não se restringe somente à análise de textos, sendo aplicada também em imagens e até em pesquisas científicas, como, por exemplo, em dados coletados em amostras biológicas oriundos de investigações sobre seres vivos. Em síntese, trata-se de um método voltado para o agrupamento de dados que compartilham características semelhantes.

No seu artigo *A Survey of Topic Modeling in Text Mining* de 2015, os autores afirmam sobre *topic modeling*:

> A principal importância da modelagem de tópicos é descobrir padrões de uso das palavras e como conectar documentos que compartilham padrões semelhantes. Assim, a ideia de modelos de tópicos é que os termos podem ser associados a documentos, e esses documentos são combinações de tópicos, em que cada tópico corresponde a uma distribuição

[4]Disponível em `https://sidra.ibge.gov.br/`. Acesso em: 22 nov. 2025.
[5]Disponível em `https://www.gov.br/conab/pt-br`. Acesso em: 22 nov. 2025.
[6]Disponível em `https://www.ibge.gov.br/estatisticas/sociais/trabalho/9662-censo-demografico-2010.html`. Acesso em: 22 nov. 2025.

> probabilística sobre as palavras. Em outras palavras, o modelo de tópicos é um modelo generativo para documentos, que especifica um procedimento probabilístico simples pelo qual eles podem ser gerados. (Alghamdi; Alfalqi, 2015, p. 1).

Assim, a modelagem de tópicos constitui uma técnica aplicada à classificação e ao agrupamento de notícias a partir da análise do conteúdo textual de suas redações. Cabe destacar que existem diversos algoritmos e implementações possíveis para essa abordagem; entretanto, todos compartilham o mesmo propósito: identificar padrões recorrentes nos textos e organizar os documentos com base nas características mais frequentes e na similaridade entre eles.

Os agrupamentos gerados pela modelagem de tópicos são, em certa medida, abstratos e não rotulados, uma vez que o processo não é orientado por categorias previamente definidas. Por esse motivo, quando o objetivo é prever explicitamente o tema ou assunto de um texto com base em uma lista pré-estabelecida de rótulos, essa técnica pode não ser a mais adequada.

4 CONCEITOS BÁSICOS

4.1 O PROCESSO *ETL*

O processo ETL (do inglês, *Extraction, Transformation and Loading*, ou em português, “Extração, Transformação e Carregamento”) é um padrão amplamente utilizado para a integração e processamento de dados. De acordo com El-Sappagh *et al.* (2011), ele é composto por 3 (três) etapas principais: a extração dos dados de diversas fontes, a transformação para adequação e padronização dos dados e o carregamento em um repositório final destinado à análise ou utilização posterior, como mostra a Figura 1.

Figura 1: O processo ETL.

Fonte: Elaborado pelo autor.

Cada etapa do processo apresenta delimitações precisas e bem estabelecidas, com funções e responsabilidades distribuídas ao longo de toda a cadeia de processamento, a saber:

- Extração (*Extraction*): é a etapa inicial do processo ETL, onde é feita a coleta dos dados a serem analisados a partir das suas respectivas fontes. Essas fontes podem variar conforme o domínio da pesquisa e incluir websites (como realizado neste trabalho), bancos de dados, livros digitalizados, anotações de campo, entre outras. De

acordo com El-Sappagh *et al.* (2011), é essencial nesta fase assegurar a integração eficiente e adequada das diversas fontes utilizadas, visando garantir a interoperabilidade e manter a consistência estrutural dos dados brutos coletados;

- Transformação (*Transformation*): é a segunda etapa do processo ETL, diretamente dependente da primeira fase. Durante a extração, os dados coletados frequentemente apresentam inconsistências ou valores inadequados, tais como valores nulos e *outliers*. Conforme apontado por El-Sappagh *et al.* (2011), a transformação envolve a limpeza e o refinamento desses dados, eliminando a possibilidade de gerar informações incorretas, incompletas, inconsistentes ou ambíguas. Tal cuidado, permite assegurar que apenas dados válidos sejam utilizados nas etapas posteriores da análise pretendida;
- Carregamento (*Loading*): é a etapa final do processo ETL, consistindo na inserção dos dados, agora extraídos e tratados, em estruturas dimensionais capazes de serem acessadas pelos usuários finais ou sistemas de aplicação. Tais estruturas podem incluir bancos de dados especializados, visualizadores de dados, APIs remotas, entre outros. Segundo El-Sappagh *et al.* (2011), essa etapa visa garantir que os dados estejam prontos para utilização prática, sendo armazenados em formatos adequados para análises futuras.

Assim, a metodologia ETL é frequentemente aplicada no contexto de integração e análise de dados, e permite estruturar o processo desde a coleta até a disponibilização das informações para interpretação. No caso de conjuntos de dados como os das notícias do site do IF Goiano, esse padrão possibilita organizar as etapas de extração, transformação e carregamento de forma sistemática, favorecendo o tratamento adequado do material coletado. A aplicação desse processo, aliada a técnicas de análise de dados e mineração de texto, permite gerar visualizações, gráficos e painéis que facilitam uma compreensão ampla e interligada dos resultados obtidos.

4.2 ROBÔS DE SOFTWARE

Robôs de *software*, também conhecidos pelo termo em inglês *bots*, são programas desenvolvidos para executar tarefas de forma automatizada, com pouca ou nenhuma

intervenção humana. Eles podem funcionar de maneira independente ou integrados a sistemas e conjuntos de ferramentas maiores, assumindo atividades que antes eram realizadas manualmente. Essa automação permite agilizar processos, reduzir erros e liberar tempo para que o trabalho humano seja direcionado a atividades mais estratégicas.

Segundo Santhanam *et al.* (2022), “em engenharia de *software*, *bots* variam desde simples *scripts* automatizados até sistemas autônomos capazes de tomar decisões”. No artigo *Bots in Software Engineering: A Systematic Mapping Study*, os autores destacam que o espectro desse tipo de software é tão amplo e diversificado que, para categorizá-lo de forma consistente, é necessário um estudo sistemático que considere três aspectos centrais: o porquê (*why*), o quê (*what*) e o como (*how*) os *bots* são utilizados na Engenharia de Software.

O *porquê* refere-se às aplicações dos *bots*, isto é, ao contexto em que são inseridos e ao problema que foram projetados para resolver. Segundo os autores, as principais aplicações incluem a manutenção e o gerenciamento automático de repositórios de software, sistemas de pergunta e resposta (*Question-Answer systems*) e o apoio ao processo de desenvolvimento de software como um todo, abrangendo todo o ciclo de vida de soluções digitais, sejam elas locais, hospedadas na nuvem ou em ambientes híbridos.

O *quê* está relacionado à tentativa de identificar os tipos de bots a partir de sua forma principal de interação com o usuário final. Entre os mais comuns, destacam-se os *chatbots* modernos, como o ChatGPT (OpenAI) e o Gemini (Google), que se comunicam por meio de mensagens de texto e, progressivamente, incorporam mídias digitais como imagens e vídeos. Também existem os bots colaborativos, projetados para executar tarefas em paralelo ao usuário, permitindo que ele avance em outras atividades enquanto o bot realiza parte do trabalho. Um exemplo desse modelo pode ser observado em agentes de IA presentes em plataformas como o Manus AI, desenvolvido pela startup chinesa Monica.

Para concluir a explicação sobre o método de investigação apresentado no artigo citado, os autores tratam do *como*, relacionando-o diretamente aos princípios de design que orientam a criação dessas ferramentas. De acordo com eles, tais princípios podem ser organizados em dois agrupamentos principais: (i) identidade do *bot* (*bot per-*

sona); e (ii) Interação Humano-Computador (IHC). Segundo os autores, “essas características focam nos aspectos sociais dos *bots* e em como eles estão sendo percebidos pelos desenvolvedores”.

4.3 WEB SCRAPING

A era do *big data* viabilizou análises mais aprofundadas e a obtenção de *insights* relevantes para empresas e organizações. Isso se tornou possível à medida que métodos e técnicas de ciência e análise de dados passaram a ser aplicados a grandes volumes de informações, muitas das quais disponíveis em fontes de dados estruturadas em páginas ou documentos HTML na web.

Para acessar e utilizar esses conteúdos, podem ser adotadas duas abordagens: (i) realizar o processo manualmente, navegando em cada fonte de interesse e salvando os dados em formatos como PDF, CSV ou planilha; ou (ii) recorrer a um programa *bot*, ou robô de software, para automatizar a extração e armazenamento para análise posterior. Evidentemente, o primeiro método torna-se inviável diante do tempo despendido, sobretudo considerando o grande volume de fontes de dados disponíveis.

Nesse contexto, o *web scraping* está diretamente associado à segunda abordagem, pois consiste na utilização de programas de computador para extrair, de forma automática e estruturada, dados provenientes de páginas da web na era do *big data*. Conforme Lotfi *et al.* (2023), essa técnica “representa uma abordagem fundamental” no tratamento de grandes volumes de dados. Além disso, os autores destacam a relevância de as organizações desenvolverem a capacidade de “minerar” informações disponíveis na *web*, de modo a extrair conhecimento útil. Tal “mineração de dados” na web, define bem o que é *web scraping*.

Ainda de acordo com Lotfi *et al.* (2023), as técnicas de *web scraping* apresentam ampla aplicabilidade, podendo ser empregadas em diferentes áreas e domínios de dados, tais como:

- Assistência médica (*healthcare*): os autores destacam que os sistemas hospitalares encontram-se atualmente altamente digitalizados, tornando o gerenciamento da

coleta de dados dos pacientes um processo “tedioso e árduo”. Nesse contexto, as técnicas de *web scraping* podem ser aplicadas para desenvolver sistemas de coleta automática de informações clínicas, como no caso de registros de pacientes com SARS-CoV-2 que visitam o hospital, permitindo a organização e disponibilização desses dados de forma autônoma para pesquisas futuras.

- Mídias sociais: além da aplicação na coleta automatizada de dados clínicos, as técnicas de *web scraping* também podem ser empregadas no aprimoramento de campanhas de marketing empresarial. Nesse caso, a ênfase recai sobre a identificação de padrões de comportamento dos clientes por meio da extração automática de publicações em redes sociais, como o Instagram, possibilitando ainda a análise de sentimentos presentes no feedback fornecido acerca dos produtos comercializados.

- Finanças: as técnicas de *web scraping* podem ser aplicadas na extração automática de textos online referentes a empresas do setor de inovação, possibilitando o treinamento de modelos de IA capazes de avaliar se essas organizações atendem a parâmetros de classificação relacionados a indicadores de inovação e ao desenvolvimento de novas soluções tecnológicas. Esses indicadores, por sua vez, podem auxiliar investidores na identificação de startups e micronegócios mais alinhados ao seu perfil, contribuindo para decisões de investimento mais assertivas.

- Marketing: além da extração e análise de dados provenientes de mídias sociais para identificar o sentimento de publicações online, as técnicas de *web scraping* podem explorar o grande volume de pegadas digitais deixadas por clientes em plataformas de comércio eletrônico. Tais informações permitem a realização de análises comportamentais e a predição da intenção de compra por meio do treinamento de modelos de aprendizado de máquina (*machine learning*). Nesse sentido, os registros de acessos a determinados produtos são utilizados como indicadores para estimar quais seriam as próximas aquisições do cliente em um período específico.

- Outras aplicações: como evidência da ampla versatilidade das técnicas de *web scraping*, Lotfi *et al.* (2023) destacam ainda sua utilização em diferentes contextos, tais como: (i) extração de informações estratégicas para setores industriais, como a indústria de minérios; (ii) desenvolvimento de assistentes voltados a designers de

moda, fornecendo dados atualizados sobre as tendências mais recentes; (iii) exploração de dados disponibilizados em plataformas online para a obtenção de informações florestais detalhadas; e (iv) geração de inferências sobre segurança no trânsito a partir da coleta de informações de sites e plataformas na Internet com o uso de bibliotecas Python.

4.4 AIRFLOW

De acordo com Singh (2019), o *Airflow* é uma plataforma destinada ao gerenciamento de tarefas em fluxos de trabalho, possibilitando organizar de forma eficiente a execução, o agendamento, a distribuição e o monitoramento dessas tarefas. Criado originalmente por engenheiros da *Airbnb*, o projeto foi posteriormente disponibilizado como código aberto e incorporado ao conjunto de softwares mantidos sob a licença *Apache*, administrada pela fundação homônima.

A plataforma disponibiliza uma API (*Application Programming Interface*, ou "Interface de Programação de Aplicação") em formato de biblioteca, compatível com diversas linguagens de programação. Por meio dessa API, podem ser definidos *scripts* que descrevem uma DAG (*Directed Acyclic Graph*, ou "Grafo Acíclico Direcionado"), estrutura central responsável por organizar os fluxos de trabalho em tarefas (*tasks*) e operadores (*operators*).

O Código 1 apresenta um exemplo de DAG escrita em Python, configurada para execução diária às 18h e composta por três tarefas ilustrativas: baixar, processar e carregar dados. Nas linhas 1 a 5, realiza-se a criação de um objeto DAG, juntamente com a definição de seu identificador, da data de início da execução e da periodicidade agendada.

```
1         with DAG(  # Criação de um objeto DAG.
2             dag_id='dag_exemplo',           # ID da DAG.
3             start_date=datetime(2024, 2, 15),  # Início de execução.
4             schedule_interval='0 18 * * *'  # Executar todos os dias às 18:00 h
5         ) as dag:
6             t1 = PythonOperator(            # Criação de uma task.
7                 task_id='baixar_dados',     # ID da task.
8                 python_callable=baixar_dados  # Função a ser executada.
9             )
10
11            t2 = PythonOperator(
12                task_id='processar_dados',
13                python_callable=processar_dados,
14                depends_on_past=True        # Depende da task anterior.
15            )
16
17            t3 = PythonOperator(
18                task_id='carregar_resultados',
19                python_callable=carregar_resultados,
20                depends_on_past=True
21            )
22
23            t1 >> t2 >> t3  # Sequência de execução das tasks da DAG.
```

Código 1: Script Python com uma DAG de exemplo.

Nas linhas 6 a 9, 11 a 15 e 17 a 21, são definidos os operadores que compõem a DAG, sendo atribuído a cada um um identificador (`task_id`) e a função Python responsável por sua execução quando a *task* é acionada. Para o segundo e o terceiro operador, especifica-se ainda que sua execução depende da finalização da task anterior, por meio do parâmetro `depends_on_past=True`, o que garante que a tarefa seguinte só seja executada após o término bem-sucedido da anterior.

A ordem de execução entre as *tasks* é definida na linha 23 por meio do operador >>, conhecido em Python como operador de deslocamento à direita. No contexto do Airflow, esse operador foi sobrecarregado (processo denominado *operator overloading*, no qual um operador assume um comportamento adicional ao originalmente definido) para representar relações de dependência entre tarefas, atuando como um `set_downstream`, isto é, indicando que a *task* posicionada à esquerda deve ser executada antes da *task* à

direita.

4.5 MINERAÇÃO DE TEXTO

A mineração de texto, ou *text mining*, constitui uma subárea da mineração de dados (*data mining*) voltada à identificação de padrões em textos não estruturados, possibilitando a extração de conhecimento relevante a partir desse material. Frequentemente, essa abordagem recorre a métodos de aprendizado de máquina (*machine learning*) para treinar modelos de inteligência artificial capazes de assimilar informações a partir de um corpus textual.

Conforme Tan (1999), o formato textual constitui a forma mais comum de armazenamento de informações nas organizações, representando cerca de 80% do conteúdo documental presente nos registros corporativos. Essa predominância torna a área de *text mining* uma das mais relevantes dentro do campo de *data mining*. Todavia, também se apresenta como uma das mais complexas, em razão da natureza despadronizada e ambígua dos textos. Essa característica demanda que a análise seja realizada de forma minuciosa e criteriosa, considerando os múltiplos significados que uma mesma sentença pode assumir em diferentes contextos.

Tan (1999) também ressalta o caráter multidisciplinar da área, destacando a participação ativa de diferentes campos do conhecimento no estudo de *text mining*. Entre as disciplinas correlatas mencionadas pelo autor estão: (i) recuperação da informação (*information retrieval*); (ii) análise de texto (*text analysis*); (iii) extração da informação (*information extraction*); (iv) agrupamento (*clustering*); (v) categorização (*categorization*); (vi) visualização (*visualization*); (vii) tecnologia de banco de dados (*database technology*); (viii) aprendizado de máquina (*machine learning*); e (ix) mineração de dados (*data mining*).

No artigo *Getting Started in Text Mining*, Cohen e Hunter (2008) destacam que os métodos de *text mining* são amplamente utilizados para favorecer a descoberta e a disseminação do conhecimento científico da literatura de biomedicina. Os autores ressaltam que as motivações para aplicar essa disciplina nesse campo são tão diversas quanto os próprios biocientistas, de modo que cada tipo de pesquisa encontra uma justificativa específica para sua utilização.

Segundo Cohen e Hunter (2008), existem pelo menos três métodos principais para a realização de mineração de texto, especialmente aplicados à biomedicina, mas também extensíveis a outras áreas do conhecimento:

- Coocorrência (*co-occurrence*): busca identificar e estabelecer correlações entre termos e conceitos que aparecem em um mesmo trecho de texto ou em posições próximas;
- Sistemas baseados em regras (*rule-based systems*): adotam uma abordagem formal fundamentada na estrutura dos idiomas, isto é, no estudo da gramática e de suas regras. Dessa forma, realizam uma análise contextual dos elementos em estudo e das relações entre eles, aplicando técnicas de linguística avançada e análise semântica;
- Métodos estatísticos ou baseados em aprendizado de máquina (*statistical or machine-learning-based*): empregam modelos de inteligência artificial que atuam como classificadores, atribuindo rótulos (*labels*) a palavras e processando árvores sintáticas completas, com o objetivo de categorizar frases e documentos.

Os autores destacam que as aplicações de *text mining* com maior usabilidade e durabilidade tendem a ser aquelas desenvolvidas não por especialistas da área, mas pelos próprios biocientistas. Isso ocorre porque os biocientistas computacionais constroem sistemas orientados à solução de problemas práticos enfrentados em seu cotidiano. Tais aplicações apresentam, em geral, três características principais:

- São específicas de um domínio (*domain-specific*), ao contrário dos sistemas produzidos por especialistas em *text mining*, que frequentemente visam apenas à publicação em conferências de linguística computacional;
- Focam na resolução de uma única atividade considerada central, diferentemente dos sistemas de especialistas que buscam integrar múltiplos métodos em uma mesma aplicação;
- Adotam uma abordagem baseada em conhecimento (*knowledge-based approach*), em contraste com a ênfase estatística predominante nos sistemas desenvolvidos por especialistas.

Dessa forma, evidencia-se a estreita relação do *text mining* com as ciências e com múltiplas áreas do conhecimento investigativo. Trata-se de um campo promissor, responsável por viabilizar tecnologias emergentes que vêm transformando a maneira como as pessoas vivem, trabalham e interagem, especialmente por meio dos modelos de inteligência artificial generativa.

4.6 ANÁLISE DE SENTIMENTO

A Análise de Sentimento constitui uma subárea da Mineração de Texto voltada ao emprego de técnicas da Ciência da Computação e da Engenharia de Software na identificação do sentimento predominante em um conteúdo textual, atribuindo-lhe um grau de positividade ou negatividade (Figueiredo *et al.*, 2018). Sua aplicação mais recorrente está na avaliação de comentários e feedbacks de clientes sobre produtos e serviços, sobretudo com a expansão das plataformas digitais de compra, nas quais cada item disponibilizado apresenta classificações e pareceres fornecidos diretamente pelos consumidores.

Trata-se, portanto, de uma ferramenta imprescindível para as empresas, especialmente por possibilitar a análise do nível de popularidade e do grau de sucesso de suas marcas no mercado global. A partir desses dados, torna-se viável estabelecer novas metas, revisar práticas comerciais, implementar ajustes em produtos e desenvolver campanhas de marketing direcionadas, entre outras ações estratégicas.

Segundo Mejova (2009), a análise de sentimento encontra-se fortemente associada a três campos centrais da computação moderna, sendo, em algumas abordagens, considerada parte deles: (i) linguística computacional; (ii) processamento de linguagem natural; e (iii) mineração de texto. A autora ressalta ainda que esse campo surgiu como uma proposta voltada à busca de respostas já recorrentes na área, tendo sido fortemente influenciado por contribuições da psicologia, em especial pelos estudos sobre o estado afetivo (*affective state*) e pela teoria da avaliação (*appraisal theory*).

O campo da análise de sentimento é intrinsecamente complexo, conforme destaca Mejova (2009). Nesse sentido, faz-se necessária a sua divisão em tarefas menores que, de forma integrada, possibilitam a realização da análise completa, permitindo a iden-

tificação e a interpretação do sentimento inferido no texto. Em linhas gerais, essas tarefas podem ser organizadas em três categorias principais, embora outras áreas e subcampos de pesquisa possam complementar essa subdivisão. São elas:

- Detecção de opinião/sentimento (*sentiment/opinion detection*): nesta etapa, os adjetivos e/ou advérbios presentes no texto são examinados a fim de identificar se a classificação deverá ser objetiva ou subjetiva. Esta última, por sua maior complexidade, demanda o emprego de técnicas mais sofisticadas no âmbito da análise de sentimento;

- Classificação de polaridade (*polarity classification*): refere-se ao processo de categorizar a opinião identificada no texto em duas polaridades de sentimento, geralmente 0 (negativo) e 1 (positivo), atribuindo-lhe um *score* (pontuação). Para isso, podem ser empregados cálculos de natureza matemática e/ou estatística que possibilitam a determinação dessa medida;

- Descoberta do alvo da opinião: nem sempre é evidente qual é o objeto a que um comentário ou opinião se refere, ou se, de fato, está relacionado ao item em análise. Mejova (2009) ressalta que, no caso de avaliações de produtos, essa identificação tende a ser mais simples, uma vez que é razoável presumir que os usuários direcionem seus comentários ao próprio produto mencionado, sendo pouco provável a inclusão de temas externos ou irrelevantes nesse tipo de ambiente.

Uma ampla gama de ferramentas e técnicas é empregada na execução dessas tarefas, salienta Mejova (2009), entre as quais se destacam o aprendizado de máquina e a marcação de classes gramaticais (*part-of-speech tagging*). Essas ferramentas fornecem a base necessária para estruturar o texto e preparar os dados para etapas mais avançadas da análise de sentimento.

Ainda segundo a autora, a metodologia para alcançar resultados consistentes envolve uma sequência de etapas complementares. O processo inicia-se com a classificação do texto por meio de técnicas como TF-IDF (*Term Frequency – Inverse Document Frequency*) e *n-grams*, prosseguindo-se, em seguida, à identificação da orientação semântica das palavras.

Essa identificação é feita em duas modalidades distintas, sendo elas: (i) pela

contagem de ocorrências de termos com polaridade previamente conhecida; (ii) utilização de documentos de treinamento com dados rotulados. Esse mesmo raciocínio é aplicado em níveis mais amplos (frases, sentenças e documentos), seguido da extração de características do objeto alvo do sentimento e, por fim, da detecção de frases comparativas.

A técnica TF-IDF (*Term Frequency – Inverse Document Frequency*) descrita acima resulta da combinação de duas abordagens consolidadas em mineração de texto: (i) *Term Frequency*, que corresponde à contagem da frequência de ocorrência de um termo em determinado documento; e (ii) *Inverse Document Frequency*, que atribui pesos menores a termos muito recorrentes e maiores a termos mais específicos. Como exemplo, preposições da língua portuguesa, como “de” e “por”, recebem menor pontuação por serem elementos morfológicos frequentes em praticamente qualquer enunciado (QAISER; ALI, 2018).

Os *n-grams* correspondem a sequências contíguas de itens, que podem ser letras, sílabas, palavras ou tokens, formando novas unidades textuais a serem analisadas posteriormente. Na abordagem de Cavnar e Trenkle (2001), um *n-gram* é definido como um recorte de n caracteres de uma cadeia textual maior, obtido a partir de subsequências contíguas do texto. Em um sentido mais amplo, *n-grams* também podem ser construídos a partir de outras unidades, como palavras, em que as sequências são formadas por grupos contíguos de tokens em vez de caracteres.

4.7 BERT

BERT, sigla para *Bidirectional Encoder Representations from Transformers*, é considerado um dos precursores dos atuais *Large Language Models* (LLMs) (Devlin *et al.*, 2019). Além de ser amplamente utilizado em tarefas de Processamento de Linguagem Natural (PLN), como a análise de sentimentos em textos, trata-se de um modelo de aprendizado de máquina não supervisionado voltado à representação e à compreensão da linguagem natural, isto é, a linguagem utilizada pelos seres humanos.

O trabalho inicial foi publicado em Devlin *et al.* (2019), por pesquisadores do departamento da *Google Research*. Nesse artigo científico, intitulado *BERT: Pre-training*

of Deep Bidirectional Transformers for Language Understanding, traduzido como BERT: Pré-treinamento de *Transformers* Bidirecionais Profundos para Compreensão de Linguagem, o modelo foi apresentado pela primeira vez como uma inteligência artificial de linguagem desenvolvida pela empresa.

De acordo com Devlin *et al.* (2019), a arquitetura do modelo é baseada em um codificador *Transformer* bidirecional de múltiplas camadas, seguindo a implementação feita por Vaswani *et al.* (2017), publicada na biblioteca *tensor2tensor* disponível no *PyPi*. Esse tipo de rede para aprendizado de máquina funciona implementando:

> Uma arquitetura de modelo que evita a recorrência e, em vez disso, depende inteiramente de um mecanismo de atenção para estabelecer dependências globais entre a entrada e a saída. O *Transformer* permite significativamente mais paralelização e pode atingir um novo estado da arte em qualidade de tradução após ser treinado por apenas doze horas em oito GPUs P100. (DEVLIN *et al.*, 2019, p. 2).

Ainda segundo Devlin *et al.* (2019), seu funcionamento pode ser dividido em pelo menos 2 (duas) etapas principais: pré-treinamento (*pre-training*) e ajustamento (*fine-tuning*), que podem ser vistas de uma forma mais visual na Figura 2:

- Pré-treinamento (*pre-training*): o modelo é pré-treinado em 2 (duas) tarefas primordiais utilizando dados não rotulados (método não-supervisionado) (1) MLM (do inglês, *Masked Language Model*, ou, “Modelo de Linguagem com Máscara”), onde uma porcentagem dos *tokens* (i.e., unidade básica de texto) é mascarada (i.e., cada *token* escolhido é substituído por uma máscara como, por exemplo, [MASK]), de forma aleatória, e depois, faz-se com que o modelo tente prever quais são esses *tokens* e (2) NSP (do inglês, *Next Sentence Prediction*, ou, “Previsão da Próxima Sentença”), a qual prepara o modelo para entender as relações entre sentenças, predizendo, de forma binária, se uma é a próxima de outra ou não. Os dados utilizados nesta etapa foram extraídos do BooksCorpus (800 milhões de palavras) e da Wikipédia em inglês (2.5 bilhões de palavras);
- Ajustamento (*fine-tuning*): finalmente, o modelo é configurado com os parâmetros obtidos da etapa de pré-treinamento, sendo atualizados de acordo com os dados rotulados gerados em cada tarefa subsequente, passando de uma para a próxima e, assim, sucessivamente. Conforme explicado pelos autores do *transformer*,

> para cada tarefa, simplesmente conecta-se as entradas e saídas específicas da tarefa no BERT e ajusta-se todos os parâmetros de ponta a ponta. Na entrada, a sentença A e a sentença B do pré-treinamento são análogas a (1) pares de sentenças na paráfrase, (2) pares de hipótese-premissa na implicação textual, (3) pares de pergunta e passagem na resposta a perguntas e (4) um par de texto-□ degenerado na classificação de texto ou rotulagem de sequência. Na saída, as representações dos *tokens* são alimentadas em uma camada de saída para tarefas de nível de *token*, como rotulagem de sequência ou resposta a perguntas, e a representação [CLS] é alimentada em uma camada de saída para classificação, como implicação textual ou análise de sentimento. (DEVLIN *et al.*, 2019, p. 5).

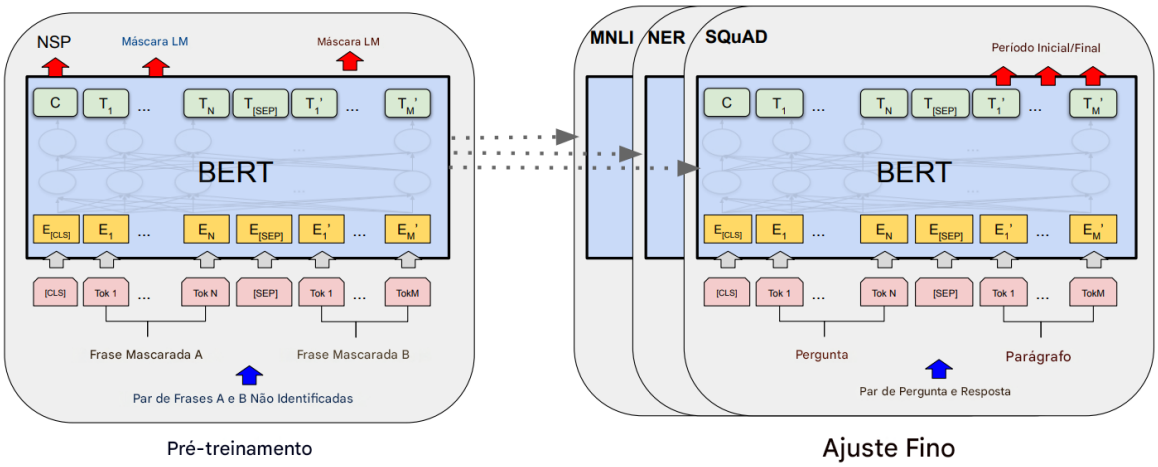

Figura 2: Procedimentos gerais de pré-treinamento e ajustamento para o BERT.

Fonte: Devlin *et al.*, 2019, p. 3.

4.8 BART

BART, sigla para *Bidirectional and Auto-Regressive Transformer* (do inglês, "Transformer Bidirecional e Auto-Regressivo"), é um modelo de aprendizado de máquina pré-treinado para tarefas de processamento de linguagem natural que adota a arquitetura *Transformer* (Lewis *et al.*, 2019). Embora compartilhe diversas características com o modelo BERT, o BART se distingue pela incorporação de um decodificador similar ao do GPT, o que lhe permite processar os textos da esquerda para a direita, combinando, assim, as abordagens bidirecional e autorregressiva em um único modelo.

O modelo foi apresentado em Lewis *et al.* (2019), do departamento de inteligência artificial da empresa Facebook à época, hoje Meta Inc., com o título em inglês,

BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, em português, algo como “BART: Pré-treinamento de Sequência-para-Sequência de Remoção de Ruído para Geração, Tradução e Compreensão de Linguagem Natural”.

Como já foi mencionado, o BART compartilha diversas características com modelos bidirecionais, como o supracitado BERT, mas também incorpora elementos fundamentais de modelos autorregressivos, como o GPT da OpenAI, responsáveis pela etapa de decodificação (ou geração) de texto. O principal avanço de sua arquitetura reside justamente na integração das abordagens bidirecional e autorregressiva em um único modelo, combinando as melhores práticas de processamento de linguagem natural disponíveis à época de seu desenvolvimento.

Segundo Lewis *et al.* (2019), o principal *insight* do modelo consistiu em combinar diferentes estratégias de pré-treinamento, integrando duas abordagens complementares:

- uma generalização do modelo *BERT*, por meio da aplicação de um *encoder* bidirecional;
- a metodologia do *GPT*, com a utilização de um decodificador esquerda-para-direita (*left-to-right*).

O seu funcionamento pode ser estratificado em 2 (duas) etapas principais, como se vê em Lewis *et al.* (2019), (1) corromper o texto com uma função arbitrária de introdução de ruído (embaralhar o seu conteúdo), e (2) aprender um modelo para reconstruir o texto original. Essas etapas são muito bem descritas pelos autores como sendo uma junção dos principais recursos oferecidos pelos outros modelos, isto é, aproveitar a interação bidirecional advinda do BERT e a capacidade de geração textual dos modelos GPT, em um único modelo, sendo que:

> Uma vantagem chave deste sistema é a flexibilidade na introdução de ruído; podem ser aplicadas transformações arbitrárias ao texto original, incluindo a alteração de seu comprimento. Avaliamos várias abordagens de introdução de ruído, encontrando o melhor desempenho tanto ao reordenar aleatoriamente a ordem das sentenças originais quanto ao usar um esquema inovador de preenchimento interno, onde trechos de texto de comprimento arbitrário (incluindo comprimento zero) são substituídos por um único token de máscara. Esta abordagem generaliza os objetivos originais de mascaramento de palavras e previsão de sentenças seguintes no BERT ao forçar o modelo a considerar mais sobre

> o comprimento total da sentença e realizar transformações em intervalos mais longos na entrada. (Lewis *et al.*, 2019, p. 1).

A fase de pré-treinamento do BART é composta por alguns procedimentos responsáveis pela corrupção do documento original em uma forma degenerada, como é explicitado a seguir e na Figura 3:

- Mascaramento de token (*token masking*): de forma similar ao BERT, alguns tokens são escolhidos aleatoriamente e substituídos por uma máscara, como [MASK];
- Deleção de token (*token deletion*): alguns tokens são escolhidos aleatoriamente para remoção, fazendo com que o modelo tenha de inferir quais são eles;
- Preenchimento de texto (*text infilling*): trechos de texto são escolhidos para serem substituídos por uma máscara de igual tamanho, ensinando o modelo a prever quantos tokens estão ausentes em um intervalo;
- Permutação de frases (*sentence permutation*): as sentenças encontradas a partir de pontos-final no documento são embaralhadas;
- Rotação de documento (*document rotation*): neste procedimento, um token é escolhido de forma uniforme e aleatória, e o documento é rotacionado para que comece com este token, o que treina o modelo para identificar onde é o início do documento.

Dessa forma, o modelo pode ser aplicado a uma ampla variedade de tarefas que envolvem compreensão e manipulação textual, com ênfase em Processamento de Linguagem Natural (PLN), como geração automática de textos, diálogo abstrato, resposta a perguntas (*Question Answering* – QA) e sumarização, isto é, a capacidade de condensar um texto extenso em uma versão mais curta, preservando as informações mais relevantes.

O processo fundamental de funcionamento do modelo se organiza em dois momentos principais: (i) no primeiro, ocorre a corrupção do documento de entrada; e (ii) no segundo, realiza-se o mapeamento do texto degradado para o texto original. A Figura 4 ilustra essa dinâmica, na qual:

> Os inputs para o codificador não precisam estar alinhados com as saídas do decodificador, permitindo transformações arbitrárias de ruído. Aqui, um documento foi corrompido substituindo trechos de texto por símbolos de máscara. O documento corrompido (à esquerda)

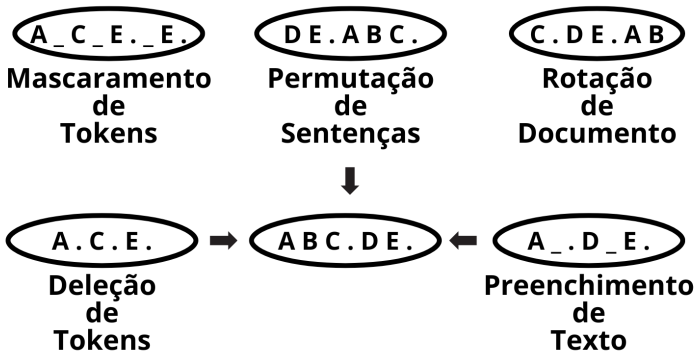

Figura 3: Transformações combináveis da fase de pré-treinamento do modelo BART para adicionar ruído à entrada.

Fonte: Lewis *et al.* (2019), p. 3. Adaptado pelo autor.

> é codificado com um modelo bidirecional, e então a probabilidade do documento original (à direita) é calculada com um decodificador autorregressivo. Para o ajuste fino, um documento não corrompido é utilizado como entrada tanto para o codificador quanto para o decodificador, e utilizamos representações do estado oculto final do decodificador. (Lewis *et al.*, 2019, p. 2).

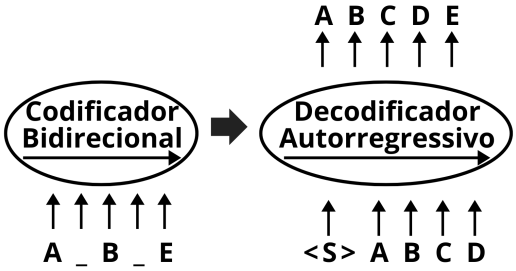

Figura 4: Codificador bidirecional (à esquerda) e decodificador autorregressivo (à direita) do modelo BART.

Fonte: Lewis *et al.* (2019), p. 2. Adaptado pelo autor.

Na Figura 5 a seguir, mostra-se a correspondência de funcionalidades encontradas entre os modelos BERT e os GPTs: o codificador bidirecional do BERT, faz com que alguns *tokens* sejam escolhidos aleatoriamente e substituídos por máscaras para posterior predição pelo modelo. Neste sentido, isto faz com que o modelo seja ideal para predição e classificação textual, e não para geração de texto.

Figura 5: Codificador bidirecional do modelo BERT.

Fonte: Lewis *et al.* (2019), p. 2. Adaptado pelo autor.

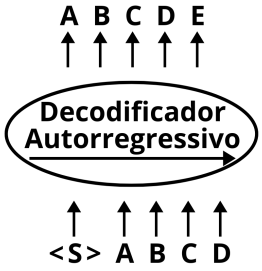

Figura 6: Decodificador autorregressivo do modelo GPT.

Fonte: Lewis *et al.* (2019), p. 2. Adaptado pelo autor.

Já na Figura 6, apresenta-se o decodificador autorregressivo dos modelos GPTs, que faz com que os *tokens* sejam preditos com base naquilo que já foi analisado previamente, ou seja, no histórico de tokens já gerados pelo modelo. Assim, essa abordagem permite que tais modelos sejam ideais para geração textual.

5 METODOLOGIA

Seguindo as etapas da metodologia ETL, os autores analisaram as redes sociais mais ativas do IF Goiano e selecionaram aquelas com maior viabilidade para a extração de dados. A partir dessa seleção, foram desenvolvidos robôs de software responsáveis por realizar coletas diárias das notícias previamente definidas, com posterior armazenamento em repositório. Em sequência, os dados passaram por um processo de limpeza e padronização e, por fim, foram inseridos em um banco de dados relacional, caracterizando a etapa final da pipeline de processamento.

Com o banco de dados já populado, contendo as notícias limpas e devidamente carregadas, iniciou-se a etapa de construção das consultas, cujo objetivo foi agrupar os dados a partir de características específicas das publicações e analisar a quantidade de acessos associada a cada grupo. Para a visualização dos resultados, recorreram-se a bibliotecas especializadas na geração de gráficos, de modo a proporcionar uma interpretação mais clara e acessível das informações extraídas.

Utilizando o modelo pré-treinado BART, foi gerado um resumo automático para cada notícia a partir de seu conteúdo integral. O objetivo consistiu em produzir versões com maior carga positiva percebida, segundo a avaliação da própria inteligência artificial, permitindo a comparação entre os resumos elaborados pela equipe de jornalismo e aqueles gerados automaticamente com base nas informações mais relevantes (*key information*). Para apoiar a análise comparativa entre os estilos de escrita humana e automatizada, foram também desenvolvidos gráficos que facilitaram a visualização das diferenças.

Como proposta de solução, foi desenvolvida uma aplicação web destinada a disponibilizar os modelos de Processamento de Linguagem Natural empregados neste trabalho em um formato de fácil acesso, eliminando a necessidade de que o usuário escreva ou desenvolva código em linguagem de programação. Os modelos disponibilizados

oferecem funcionalidades de análise de sentimento e reescrita automática de qualquer corpus textual submetido à aplicação. O repositório utilizado como fonte desses modelos de IA foi a *Hugging Face*, uma iniciativa global da comunidade que busca facilitar o acesso e a utilização desses componentes de software pelo público em geral.

6 IMPLEMENTAÇÃO

A extração dos dados foi iniciada com uma exploração preliminar do site do IF Goiano, a fim de compreender a estrutura e a organização das notícias publicadas na plataforma. A Tabela 1 apresenta os principais dados das publicações identificados no site, acompanhados de uma breve definição.

6.1 EXTRAÇÃO DAS NOTÍCIAS

6.1.1 Exploração do Site

Inicialmente, foi realizada uma exploração do site do IF Goiano, a fim de entender como se dá a estruturação e a organização das notícias publicadas na plataforma. A Tabela 1 apresenta quais são os principais dados das publicações no site do IF Goiano e uma breve definição.

Tabela 1: Domínio de dados de cada notícia publicada no site

Dado	Definição
Título	Assunto central da notícia
Descrição	Resumo do conteúdo redigido pela equipe de jornalismo
Timestamp de publicação	Data-hora em que a notícia foi publicada no site
Quantidade de acessos	Total de visitas à página da notícia

Fonte: Elaborado pelo autor.

Ao examinar o conteúdo do corpo de cada notícia, foram identificados outros dados relevantes que poderiam influenciar a análise da quantidade de acessos. Entre eles, destacam-se a presença ou ausência de imagem de perfil, o número total de imagens no corpo da notícia, o período do dia em que a publicação foi feita (manhã, tarde ou noite), o dia da semana correspondente, entre outros.

6.1.2 Desenvolvimento do Robô de Software

Foi desenvolvido um robô de software utilizando a biblioteca Selenium para acessar o site do IF Goiano[7] e coletar os dados da publicação em “Destaque” e das três “Últimas Notícias” (veja Figura 7 e Figura 8, respectivamente).

O robô foi programado para, em seguida, acessar a página redirecionada pelo link “ACESSE A LISTA COMPLETA”, localizado no canto inferior direito da página inicial, dentro da seção “Notícias Anteriores” (veja Figura 9), visitar cada notícia individualmente (veja Figura 10) e coletar seus dados principais.

Figura 7: Notícia de destaque no site do IF Goiano.

Fonte: Elaborado pelo autor.

Após a coleta, as notícias passaram por uma etapa de pré-processamento vol-

[7]Endereço disponível em `https://www.ifgoiano.edu.br`.

Figura 8: Últimas 3 (três) notícias publicadas no site do IF Goiano.

Fonte: Elaborado pelo autor.

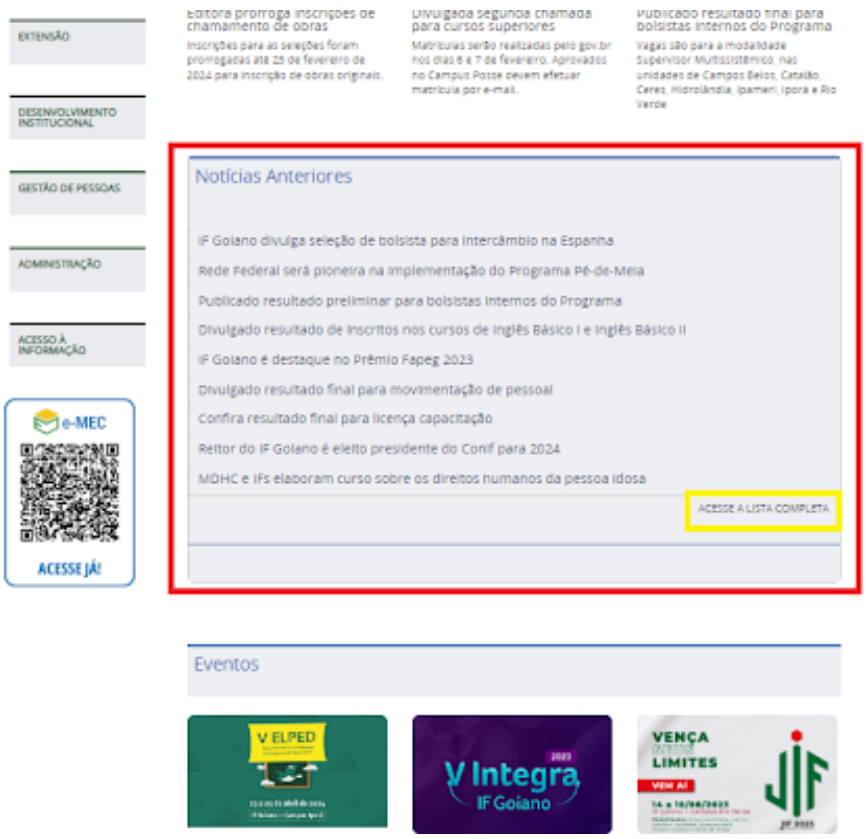

Figura 9: Notícias anteriores no site do IF Goiano.

Fonte: Elaborado pelo autor.

Figura 10: Lista com todas as notícias anteriores ao clicar em ”ACESSE A LISTA COMPLETA”no site do IF Goiano.

Fonte: Elaborado pelo autor.

tada à remoção de valores nulos e *outliers*, isto é, registros de acesso significativamente discrepantes em relação ao restante do conjunto. Concluída essa fase, os dados foram exportados para um arquivo CSV, que serviu de base para a etapa de carga no banco de dados.

```
1  class Bot:
2      def __init__(self):
3          options = ChromeOptions()
4          prefs = {
5              'profile.managed_default_content_settings.images': 2,    # Desabilitar imagens.
6              'intl.accept_languages': 'en,en_US'                      # Definir inglês como
       idioma do navegador.
7          }
8          options.add_experimental_option('prefs', prefs)
9          options.add_argument('--headless')  # Executar sem abrir o navegador.
10
```

Código 2: Definição da classe `Bot` com as principais operações do robô de coleta. Parte 1

No Código 2, na linha 1, o robô foi implementado a partir de uma classe Python denominada `Bot`. Inicialmente, algumas configurações essenciais foram estabelecidas. Na linha 5, desabilitou-se o carregamento de imagens no navegador, medida que evita sobrecarga desnecessária durante a coleta.

Na linha 6, definiu-se o idioma inglês como padrão do navegador, facilitando o acesso aos dados pelo *bot*. Por fim, na linha 9, configurou-se a execução em modo “headless”, permitindo que o robô funcione sem abrir uma janela do navegador, isto é, sem interface gráfica.

```
1         self._driver = webdriver.Chrome(
2             service=ChromeService(
3                 ChromeDriverManager(
4                     chrome_type=ChromeType.CHROMIUM
5                 ).install()
6             ),
7             options=options
8         )
9         self._driver.implicitly_wait(5)
10        self._post_counter = 0
11        self._data_dict = dict(
12            titulo=list(),
13            descricao=list(),
14            data_hora=list(),
15            dia_semana=list(),
16            periodo_dia=list(),
17            imagens=list(),
18            acessos=list(),
19            acessos_medio_hora=list(),
20            tem_imagem_perfil=list()
21        )
22        print('----- Bot is running -----')
23
```

Código 3: Definição da classe `Bot` com as principais operações do robô de coleta. Parte 2

No Código 3, entre as linhas 1 e 8, definiu-se o navegador `Chromium` para a execução do robô. Trata-se de uma versão gratuita e de código aberto, que oferece vantagens importantes para tarefas de *web scraping*, como maior estabilidade, menor consumo de recursos, controle operacional ampliado e suporte nativo ao modo de execução *headless* (sem interface gráfica).

Em seguida, entre as linhas 11 e 21, estruturou-se a coleção de dados responsável por armazenar as informações das notícias à medida que eram coletadas. Essa estrutura contempla o domínio de dados apresentado na Tabela 1, além de variáveis derivadas obtidas a partir dele, como `dia_semana` e `periodo_dia`.

```
1     def get_posts(self):
2         exec_start = datetime.now()
3         try:
4             self._driver.get(IFGOIANO_HOME)
5             # Obtém os dados da notícia em destaque.
6             featured_post_box = self._driver.find_element(By.CLASS_NAME, 'manchete-texto-
      lateral')
7             featured_post_url = featured_post_box.find_element(By.TAG_NAME, 'a').get_attribute
      ('href').strip()
8             post_has_profile_image = len(featured_post_box.find_elements(By.TAG_NAME, 'img'))
      == 1
9             self._driver.execute_script(f"window.open('{featured_post_url}', 'new page');")
10            self._driver.switch_to.window(self._driver.window_handles[1])
11            if post_has_profile_image:
12                self._data_dict['tem_imagem_perfil'].append('sim')
13            else:
14                self._data_dict['tem_imagem_perfil'].append('nao')
15            self._extract_post_data()
16            self._driver.close()
17            self._show_post_info()
18            self._driver.switch_to.window(self._driver.window_handles[0])
19
```

Código 4: Método `get_posts` da classe `Bot`. Parte 1

O método principal desta classe foi denominado `get_posts` e está apresentado no Código 4. É nesse método que se encontra a lógica central de coleta das notícias. Na linha 4, o navegador é direcionado para o endereço do IF Goiano em `https://ifgoiano.edu.br/` por meio do método `get`. Em seguida, na linha 6, obtém-se o componente HTML correspondente ao conteúdo da notícia em destaque, que serve como ponto inicial para o processo de extração dos dados.

Na linha 7, extrai-se o link para a notícia em destaque, permitindo que ele seja aberto em uma nova aba do navegador. Na linha 8, verifica-se se a notícia possui imagem de perfil, contabilizando a quantidade de *tags* `img` presentes no componente correspondente. Nas linhas 9 e 10, o link obtido na linha 7 é aberto em uma nova aba, que passa a receber o foco do robô por meio da chamada ao método `switch_to.window`. Na linha 15, os dados da notícia são extraídos pelo método interno `_extract_post_data` da classe

`Bot`. Em seguida, na linha 16, a aba é fechada e, na linha 18, o foco é retomado na aba principal do navegador, permitindo a continuidade do processo de coleta.

O método `_extract_post_data`, pertencente à classe `Bot`, foi desenvolvido para tirar proveito da estrutura relativamente uniforme das notícias, permitindo a criação de um algoritmo generalista capaz de extrair dados de qualquer notícia cujo link esteja aberto em uma aba do navegador. Essa abordagem possibilita maior reaproveitamento de código e reduz o acoplamento entre os diferentes métodos da classe `Bot`. O código correspondente se encontra na seção de apêndices.

```
1            # Obtém os dados das notícias restantes em IFGOIANO_HOME.
2            box_secondary_posts = self._driver.find_element(By.CLASS_NAME, 'chamadas-
    secundarias')
3            secondary_posts = box_secondary_posts.find_elements(By.TAG_NAME, 'div')
4            for secondary_post in secondary_posts:
5                post_has_profile_image = len(secondary_post.find_elements(By.TAG_NAME, 'img'))
     == 1
6                secondary_post_url = secondary_post.find_element(By.TAG_NAME, 'a').
    get_attribute('href').strip()
7                self._driver.execute_script(f"window.open('{secondary_post_url}', 'new page');
    ")
8                self._driver.switch_to.window(self._driver.window_handles[1])
9                if post_has_profile_image:
10                   self._data_dict['tem_imagem_perfil'].append('sim')
11               else:
12                   self._data_dict['tem_imagem_perfil'].append('nao')
13               self._extract_post_data()
14               self._driver.close()
15               self._driver.switch_to.window(self._driver.window_handles[0])
16               self._show_post_info()
17
```

Código 5: Método `get_posts` da classe `Bot`. Parte 2

Dando continuidade à apresentação das partes mais relevantes do robô de software desenvolvido para extrair os dados das notícias, o Código 5 apresenta a segunda etapa desse processo. Nesse momento, realizou-se a coleta dos dados das demais notícias exibidas na página principal do site, classificadas pelo próprio portal como **Últimas Notícias**.

Nas linhas 2 e 3 do Código 5, obtém-se a lista que contém todos os componentes HTML correspondentes ao conteúdo de cada uma das últimas notícias do site. Em seguida, na linha 4, realiza-se a iteração sobre essa lista utilizando uma estrutura comum na linguagem Python, o laço `for` com o operador `in`.

Finalmente, realizam-se os mesmos passos descritos anteriormente para a extração dos dados da notícia em destaque. Assim: (i) na linha 5, verifica-se se a notícia possui imagem de perfil; (ii) na linha 6, obtém-se o link para acessar o conteúdo completo da notícia; (iii) nas linhas 7 e 8, abre-se esse link em uma nova aba, que então recebe o foco do robô; (iv) nas linhas 9 a 13, efetuam-se a extração e o registro dos dados da notícia; e (v) por fim, fecha-se a aba e retorna-se à página principal.

A extração dos dados das demais notícias, isto é, aquelas presentes na página acessada ao clicar em “ACESSE A LISTA COMPLETA”, seguiu uma abordagem essencialmente idêntica à descrita acima. Assim, para fins de objetividade, essa parte do código não será apresentada nesta seção. Contudo, todo o código do *bot* será disponibilizado integralmente na seção de apêndices deste trabalho.

6.1.3 Agendamento da execução com Airflow

O robô de software desenvolvido foi agendado em uma DAG na plataforma Airflow, para execução diária ao longo de um período de 19 (dezenove) dias. A cada execução, o robô coletou as notícias publicadas no site do IF Goiano, pré-processou os dados obtidos e os exportou em formato CSV, consolidando a base de dados utilizada neste projeto.

O Código 6 apresenta a estrutura da DAG responsável pelo agendamento do robô de software encarregado da coleta das notícias no site do IF Goiano. Nas linhas 2 a 6, são definidos os argumentos padrão da DAG, incluindo parâmetros como o número máximo de tentativas em caso de falha, estabelecido em cinco execuções. Em seguida, na linha 7, especifica-se a data de início do agendamento, configurada para 26 de janeiro de 2023.

```
1  if __name__ == '__main__':
2      dag_args = {
3          'owner': 'Rafael',
4          'retries': 5,
5          'retry_delay': timedelta(minutes=5)
6      }
7      start_date = datetime(2023, 1, 26)
8      with DAG(
9          dag_id='ifgoiano_site_dag',
10         description='DAG para coletar dados das notícias do IF Goiano DAG for getting and
       processing data of publications from www.ifgoiano.edu.br',
11         default_args=dag_args,
12         start_date=start_date,
13         end_date=start_date + timedelta(days=19),
14         schedule='0 21 * * *',
15         catchup=False
16     ) as dag:
17         get_posts = PythonOperator(
18             task_id='get_posts',
19             python_callable=Bot().get_posts
20         )
21         get_posts
22
```

Código 6: DAG responsável por automatizar a execução do robô de software.

A partir da linha 8, emprega-se a sintaxe do bloco `with` em Python para instanciar um objeto da classe `DAG`, disponibilizada pela biblioteca `apache-airflow`[8]. O uso desse bloco garante que, ao término de sua execução, quaisquer conexões ou recursos eventualmente abertos durante a criação do objeto sejam encerrados de forma automática e adequada.

Entre os argumentos mais comuns de uma DAG, destacam-se o `dag_id`, utilizado para sua identificação, a descrição textual, os argumentos padrão, as datas de início e término da execução, além da expressão CRON, responsável por definir a periodicidade das execuções (como diária, semanal, entre outras). Outro parâmetro relevante é o `catchup`, que especifica se o Airflow deve executar retroativamente todas as instân-

[8]Acesse o conteúdo da biblioteca em `https://pypi.org/project/apache-airflow/`.

cias previstas entre o `start_date` e o momento atual.

Em seguida, nas linhas 17 a 20, foi instanciado um objeto `PythonOperator`, cuja função é definir uma tarefa Python dentro da DAG. Nesse operador, atribuiu-se o `task_id` “get_posts” e indicou-se como função de execução o método `get_posts` da classe `Bot`, previamente descrita nas seções anteriores.

Assim, o operador torna-se a entidade responsável por invocar o método especificado conforme o agendamento definido para a DAG. Por fim, na linha 21 ocorre a vinculação efetiva da tarefa ao fluxo de execução, sendo suficiente, segundo a sintaxe do Airflow, apenas referenciar o objeto do operador.

6.2 TRANSFORMAÇÃO DAS NOTÍCIAS

6.2.1 Limpeza dos Dados

Os dados das notícias coletadas foram submetidos a um processamento preliminar para a remoção de registros com valores nulos e/ou com número de acessos considerados discrepantes. Em alguns casos, notícias mais antigas apresentavam volumes de acesso muito superiores aos das publicações recentes, o que poderia comprometer a consistência da análise caso não fosse devidamente tratado.

O Código 7 descreve o trecho responsável pela etapa de limpeza dos dados. Na linha 7, é realizada a listagem de todos os arquivos CSV gerados na extração das notícias do site do IF Goiano. Em seguida, na linha 8, um laço `for` percorre cada um desses arquivos, permitindo o processamento individual de cada conjunto de dados.

```
1   import os, pandas, scipy, numpy
2
3   SOURCE_PATH = ''
4   TARGET_PATH = ''
5
6   if __name__ == '__main__':
7       files = os.listdir(SOURCE_PATH)
8       for f in files:
9           csv = pandas.read_csv(os.path.join(SOURCE_PATH, f))
10          csv.dropna(inplace=True)
11          csv_zscore = scipy.stats.zscore(csv['acessos'])
12          csv_zscore_abs = (numpy.abs(csv_zscore) < 3)
13          csv = csv[csv_zscore_abs]
14          csv.to_csv(os.path.join(TARGET_PATH, f), index=False)
15
```

Código 7: Script Python apresentando como a limpeza dos valores nulos e outliers foi feita.

Na linha 9, o conteúdo de cada arquivo é lido e armazenado em um *data frame* da biblioteca `pandas`. Logo após, na linha 10, aplica-se o comando `dropna`, que remove todas as linhas que contenham ao menos um valor nulo em qualquer coluna. Esse passo garante que apenas registros completos sigam para as etapas seguintes.

Entre as linhas 11 e 13, ocorre a identificação e exclusão dos *outliers* com base no Z-Score, utilizando um limite de corte absoluto igual a 3. Esse procedimento permite eliminar registros com número de acessos significativamente discrepantes, tanto para cima quanto para baixo, em relação à distribuição geral. Por fim, na linha 14, os dados tratados são exportados para um novo diretório, onde ficam prontos para serem carregados no banco de dados do projeto.

A função `scipy.stats.zscore` do módulo `stats` da biblioteca `scipy` permite calcular, de forma direta, o Z-Score de cada valor presente na coluna `'acessos'` do *data frame*. Esse procedimento transforma os valores brutos em uma escala padronizada, indicando quantos desvios padrão cada valor está distante da média da distribuição.

Na prática, o Z-Score é utilizado para identificar valores discrepantes (*outliers*), uma vez que, em distribuições aproximadamente normais, a maior parte dos dados tende a se concentrar em torno da média. Valores com Z-Scores muito altos ou muito baixos (por

exemplo, maiores que 3 ou menores que -3) são considerados fora da curva e, portanto, são candidatos à remoção durante a limpeza dos dados.

Essa etapa foi essencial para garantir a consistência do conjunto de dados utilizado na análise, evitando distorções nas métricas geradas. Para identificar os *outliers*, adotou-se como critério a distância em relação à média, utilizando o cálculo do Z-Score com base no desvio padrão da distribuição. Valores muito acima ou muito abaixo desse limite foram descartados, assegurando maior confiabilidade nos resultados obtidos.

6.2.2 Classificação das Notícias utilizando ChatGPT

A classificação das notícias teve como base a identificação do assunto principal abordado em cada publicação, a partir da análise do conteúdo textual presente no título e na descrição. Em colaboração com a equipe de jornalismo do IF Goiano – Campus Morrinhos, foram definidos os seguintes temas:

- **Ações de Extensão:** as ações de extensão abrangem os conteúdos relacionados à extensão, sendo publicações sobre projetos de extensão, programas, arte e cultura, eventos, estágio (convênios com empresas e seleção de estágio), emprego, egressos e parcerias com instituições, empresas e comunidade externa;
- **Campanhas:** ações planejadas de divulgação sobre diversos assuntos de interesse institucional e social;
- **Comunicados Oficiais:** os comunicados oficiais são conteúdos de caráter informativo emitidos pela instituição sobre diversos assuntos institucionais de conhecimento público;
- **Editais:** ato administrativo, escrito e oficial para divulgar normas sobre processos específicos. O edital é utilizado por diversos setores do IF Goiano, inclusive para promover o conhecimento do público interessado nos processos seletivos;
- **Processos Seletivos:** procedimentos utilizados para ranquear e classificar candidatos em seleções de diversos tipos, com destaque para os processos de seleção de ingresso nos cursos.

A abordagem inicial adotada consistiu na aplicação da técnica de mineração de texto denominada modelagem de tópicos (*topic modeling*), empregada para agrupar automaticamente documentos com base em similaridades textuais, com vistas a identificar os temas principais de cada notícia publicada no site do IF Goiano.

A modelagem de tópicos é um método não supervisionado, isto é, dispensa rótulos ou categorias previamente definidas e busca identificar padrões e estruturas recorrentes no conteúdo, de modo a formar agrupamentos naturais entre os textos.

No entanto, justamente por se tratar de uma técnica não supervisionada, ela identifica temas diferentes dos previamente definidos em conjunto com os jornalistas do IF Goiano, resultando em uma classificação divergente daquela prevista nos objetivos do projeto. Dessa forma, a técnica desconsidera a categorização temática estabelecida anteriormente, como ações de extensão e campanhas, e constrói sua própria classificação dos documentos.

Como é demonstrado na Figura 11, a aplicação da técnica sobre as notícias extraídas produziu agrupamentos baseados exclusivamente na similaridade e na ocorrência de termos e palavras, destacando alguns deles como representativos de determinados temas. Por exemplo, no primeiro agrupamento sobressaiu o termo “outubro”, enquanto no segundo prevaleceu o termo “edital”, entre outros. Esse procedimento, contudo, não gerou categorias coerentes, resultando em uma segmentação pouco alinhada à estrutura esperada para a análise.

Para contornar a limitação da abordagem anterior, adotou-se uma estratégia alternativa capaz de, a partir da lista de temas previamente definidos e de suas respectivas descrições, identificar o assunto mais provável de cada notícia com base em seu conteúdo textual. O objetivo foi direcionar a classificação de acordo com o significado efetivo da publicação, e não apenas com base em padrões de palavras recorrentes.

Nesse contexto, optou-se pela utilização de uma LLM (*Large Language Model*), especificamente o modelo ChatGPT 3.5, que demonstrou eficácia na classificação das notícias. Diferentemente dos métodos tradicionais de agrupamento, esse tipo de modelo é capaz de interpretar a linguagem natural em maior profundidade, considerar o contexto da notícia e indicar uma categoria mais adequada e compreensível para os leitores humanos.

O funcionamento desses modelos baseia-se em interações mediadas por *prompts*,

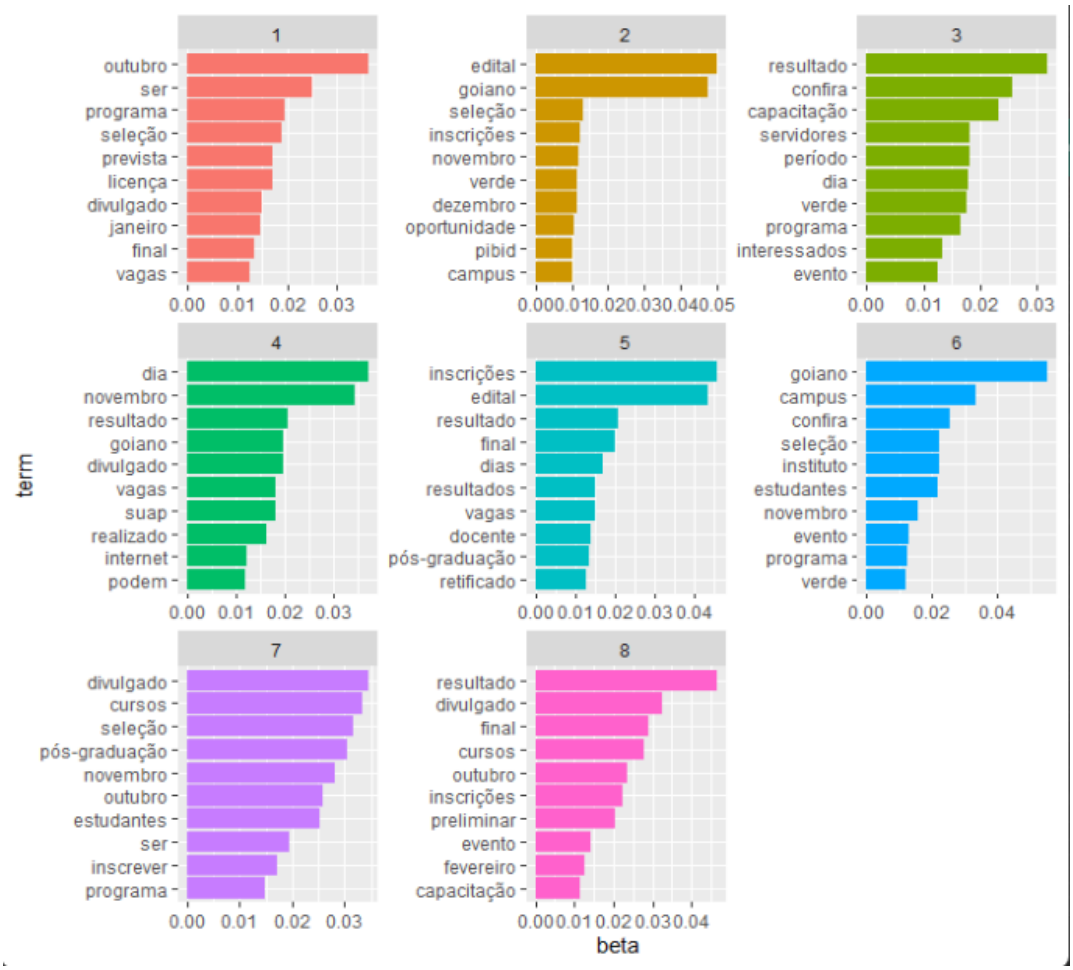

Figura 11: Modelagem de tópicos aplicada às notícias extraídas do site do IF Goiano, agrupando-as em pelo menos 8 (oito) tópicos/-categorias.

Fonte: Elaborado pelo autor.

isto é, comandos, perguntas ou instruções fornecidas pelo usuário. Em muitos casos, a formulação adequada desses prompts é essencial para obter respostas coerentes e alinhadas aos objetivos do processo, o que evidencia a importância da aplicação de técnicas de engenharia de *prompt*.

Para aumentar a precisão da classificação e reduzir ambiguidades entre os temas, foi necessário preparar cuidadosamente a entrada fornecida à IA, incluindo a definição detalhada de cada categoria existente. Esse procedimento teve como finalidade orientar o modelo na escolha mais adequada e minimizar erros de interpretação. Um exemplo de *prompt* utilizado nessa etapa é apresentado na Figura 12.

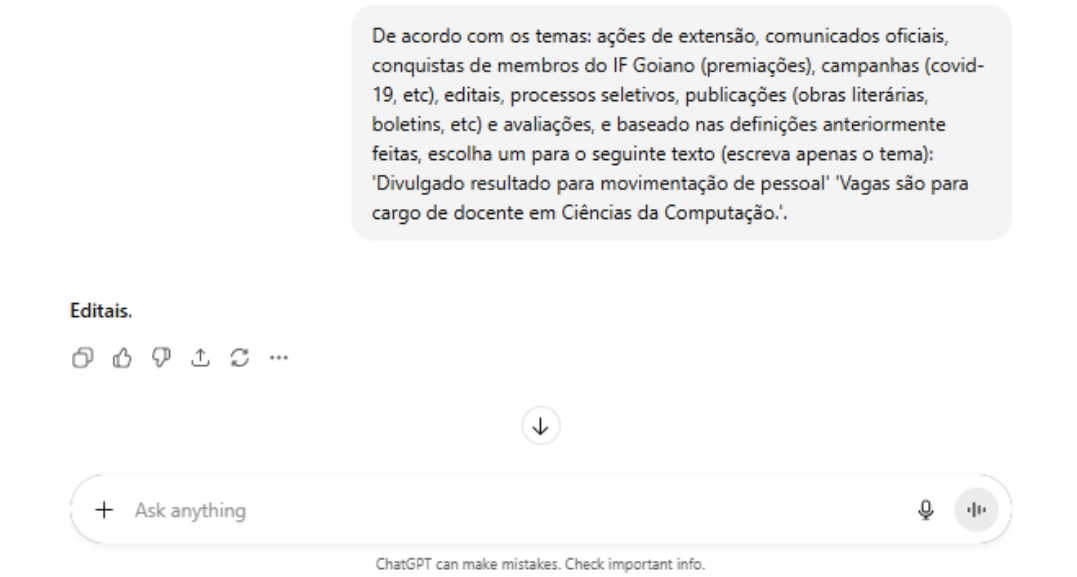

Figura 12: Interação com ChatGPT através de *prompt* contendo contexto.

Fonte: Elaborado pelo autor.

Para viabilizar o uso do ChatGPT no processo de classificação, foi desenvolvido um *script* em Python utilizando a biblioteca `openai`[9], que possibilita a interação com o modelo de IA por meio de API. Esse *script* acessa os dados de cada notícia armazenada na base, recupera título e descrição, envia essas informações ao ChatGPT e recebe como retorno a categoria considerada mais apropriada. A resposta é então associada à respectiva notícia e armazenada, juntamente com os demais campos, em um novo arquivo CSV.

No Código 8, apresenta-se um trecho desse procedimento. A etapa inicial do *script* é bastante semelhante ao que já foi descrito anteriormente neste projeto. Na linha

[9]Acesse o conteúdo da biblioteca em `https://pypi.org/project/openai/`.

20, o *script* recupera o título e a descrição diretamente de um arquivo CSV lido com a biblioteca `pandas`. Em seguida, na linha 21, ocorre o envio da solicitação ao modelo de LLM por meio do método `create`, associado ao objeto `ChatCompletion`, que inicia a interação com a API da OpenAI. O resultado da classificação é acessado na linha 25 e armazenado em um *data frame*, sendo posteriormente exportado em formato CSV.

```
1  import pandas as pd
2  import openai
3  import os
4
5  openai.api_key = os.getenv("OPENAI_API_KEY")
6
7  SOURCE_PATH = ''
8  TARGET_PATH = ''
9
10 PROMPT = (
11     "De acordo com os temas: ações de extensão, comunicados oficiais, "
12     "conquistas de membros do IF Goiano (premiações), campanhas (covid-19, etc), "
13     "editais, processos seletivos, publicações (obras literárias, boletins, etc) "
14     "e avaliações, e baseado nas definições anteriormente feitas, escolha um para "
15     "o seguinte texto (escreva apenas o tema):\n\n\"{texto}\""
16 )
17
18 def classify_subjects(df: pd.DataFrame) -> pd.DataFrame:
19     for idx, row in df.iterrows():
20         texto = f"{row['titulo']} - {row['descricao']}"
21         response = openai.ChatCompletion.create(
22             model="gpt-3.5-turbo",
23             messages=[{"role": "user", "content": PROMPT.format(texto=texto)}]
24         )
25         df.at[idx, "subject"] = response.choices[0].message.content.strip()
26     return df
27
28
29 if __name__ == '__main__':
30     files = os.listdir(SOURCE_PATH)
31     for f in files:
32         csv = pd.read_csv(f)
33         csv = classify_subjects(csv)
34         csv.to_csv(os.path.join(TARGET_PATH, f), index=False)
```

Código 8: Script Python para realizar a classificar do assunto das notícias através da API do ChatGPT.

6.2.3 Carga no Banco de Dados

Após a etapa de limpeza e classificação das notícias, todo o conjunto de dados foi carregado em um banco de dados local utilizando o *SQLite3*. O *SQLite* é um sistema de banco de dados relacional de código aberto, amplamente reconhecido por sua leveza, sem abrir mão das funcionalidades essenciais de um Sistema Gerenciador de Banco de Dados (SGBD) tradicional (LV JUNYAN *et al.*, 2009).

De acordo com LV JUNYAN *et al.* (2009), o *SQLite* apresenta características que o tornam particularmente vantajoso em aplicações de pequeno e médio porte. Entre os principais pontos positivos, destacam-se:

- **Transações ACID** (do inglês, *Atomicity, Consistency, Isolation, and Durability* – "Atomicidade, Consistência, Isolamento e Durabilidade"), que asseguram a integridade dos dados mesmo em situações de falha ou interrupção;
- **Zero configuração**, eliminando a necessidade de instalação ou configuração prévia do sistema, o que facilita sua adoção em diferentes ambientes;
- **Armazenamento em arquivo único**, possibilitando que todo o banco de dados seja mantido em um único local, tornando o sistema portátil e de fácil integração com outras aplicações.

```python
import sqlite3



if __name__ == '__main__':
    with sqlite3.connect('database.db') as conn:
        pass
```

Código 9: Script Python para se conectar um banco de dados SQLite.

No Código 9, é mostrado como é realizada a conexão com um banco de dados SQLite3. Para isso, é necessário importar o módulo `sqlite3`, que faz parte da biblioteca padrão do Python, e utilizar o método `connect`, passando como argumento o caminho para o arquivo com extensão `.db`. A utilização da cláusula `with` garante que a conexão

seja automaticamente encerrada após o término do bloco, por meio da chamada implícita ao método `close` do objeto `conn`.

Caso o arquivo informado ainda não exista, o próprio comando `connect` se encarrega de criá-lo e inicializá-lo com a estrutura padrão de um banco de dados SQLite3, o que torna o processo mais simples e mais prático, sem necessidade de uma configuração adicional. Assim, todo o código de manipulação de dados será escrito dentro do bloco do `with`, isto é, entre `with sqlite3.connect('database.db') as conn` e o encerramento do bloco.

A partir desse ponto, para permitir a persistência dos dados das notícias, criou-se a tabela `noticia`, conforme apresentado no Código 10. Essa tabela é responsável por armazenar todos os atributos relevantes extraídos ou derivados das publicações coletadas no site do IF Goiano.

Entre as primeiras colunas, destacam-se: (i) `titulo`, que contém o assunto principal da notícia; (ii) `descricao`, com o resumo redigido pela equipe de jornalismo; (iii) `data_hora`, representando o *timestamp* da publicação; (iv) `data_coleta`, representando o *timestamp* da coleta; (v) `acessos`, que registra o número total de visualizações da página até a data da coleta; e (vi) `assunto`, referente à classificação temática atribuída pela IA.

```python
import sqlite3

with sqlite3.connect("database.db") as conn:
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS noticia (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            titulo TEXT,
            descricao TEXT,
            data_hora DATETIME,
            data_coleta DATETIME,
            acessos INTEGER,
            assunto TEXT,
            sentimento_titulo REAL,
            sentimento_descricao REAL,
            resumo_gerado TEXT,
            quantidade_imagens INTEGER,
            possui_imagem_perfil INTEGER,
            dia_semana TEXT,
            periodo TEXT
        );
    ''')
    conn.commit()
```

Código 10: Script Python para criar a tabela noticia no banco de dados SQLite.

Além dessas, outras colunas foram adicionadas para enriquecer a análise posterior: (vi) `sentimento_titulo` e (vii) `sentimento_descricao`, com a polaridade emocional extraída de cada texto por meio do modelo BERT; (viii) `resumo_gerado`, contendo uma versão condensada da notícia produzida com o modelo BART; (ix) `quantidade_imagens`, que indica o número de imagens presentes na publicação; (x) `possui_imagem_perfil`, um campo booleano que identifica a presença de imagem de capa; e, por fim, (xi) `dia_semana` e (xii) `periodo`, que registram, respectivamente, o dia da semana e o momento do dia em que a notícia foi publicada.

Após a criação da tabela, procedeu-se à sua alimentação com os dados provenientes dos arquivos CSV gerados na etapa inicial, utilizando-se o comando SQL `INSERT` conforme exemplificado no Código 11. Para esse procedimento, adotou-se um bloco `with`, responsável por estabelecer a conexão com o banco de dados SQLite e assegurar o encerramento adequado dessa conexão ao término da execução.

Na sequência, implementou-se um laço `for` para percorrer todos os arquivos localizados no diretório de saída da etapa de extração, aplicando-se um filtro que restringia o processamento exclusivamente a arquivos com a extensão `.csv`.

```python
import pandas as pd
import sqlite3
import os

SOURCE_PATH = ""  # Deixado em branco para fins de ilustração.

with sqlite3.connect("database.db") as conn:
    cursor = conn.cursor()
    for arquivo in os.listdir(SOURCE_PATH):
        if not arquivo.endswith(".csv"):
            continue
        df = pd.read_csv(os.path.join(SOURCE_PATH, arquivo))
        for _, row in df.iterrows():
            cursor.execute('''
                INSERT INTO noticia (
                    titulo,
                    descricao,
                    data_hora,
                    data_coleta,
                    acessos,
                    assunto,
                    sentimento_titulo,
                    sentimento_descricao,
                    resumo_gerado,
                    quantidade_imagens,
                    possui_imagem_perfil,
                    dia_semana,
                    periodo
                ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            ''', (
                row['titulo'],
                row['descricao'],
                row['data_hora'],
                row['data_coleta'],
                row['acessos'],
                row['assunto'],
                row['sentimento_titulo'],
                row['sentimento_descricao'],
                row['resumo_gerado'],
                row['quantidade_imagens'],
                row['possui_imagem_perfil'],
                row['dia_semana'],
                row['periodo']
            ))
    conn.commit()
```

Código 11: Script Python para inserir as notícias no banco de dados SQLite.

Para cada arquivo processado, os dados foram carregados por meio da biblioteca `pandas`, sendo posteriormente convertidos em um *data frame*. Em seguida, um segundo laço percorreu o *data frame* linha a linha, extraindo os valores das colunas correspondentes e vinculando-os aos respectivos campos da tabela `noticia`.

Para cada registro, executou-se o comando SQL `INSERT INTO`, o que possibilitou a inserção dos dados estruturados no banco. Por fim, a chamada ao método `commit` assegurou a gravação definitiva de todas as inserções realizadas ao longo da execução do script.

6.3 ANÁLISE DOS DADOS

A etapa de análise teve como propósito identificar e extrair informações relevantes a partir do conjunto de dados previamente coletado, processado e armazenado no banco SQLite. Essa etapa foi estruturada em dois eixos centrais: (i) a análise quantitativa de acessos, realizada por meio de agrupamentos baseados em características específicas das notícias; e (ii) a análise de sentimento, conduzida a partir da comparação entre o texto produzido manualmente pela equipe de jornalismo e o resumo gerado automaticamente por modelos de linguagem.

6.3.1 Análise da quantidade de acessos das notícias

Após a inserção dos dados no banco, procedeu-se à Análise Exploratória dos Dados (do inglês, *Exploratory Data Analysis* – EDA), com o intuito de identificar padrões e possíveis relações entre o número de acessos das notícias e suas características associadas, como, por exemplo, a quantidade de imagens presentes em cada publicação. Para a condução dessa etapa, recorreram-se principalmente às bibliotecas `pandas` e `numpy`, em razão da eficiência e da praticidade que oferecem para a manipulação e análise de dados.

6.3.1.1 Pelo assunto e pela data da coleta

O agrupamento das notícias por assunto possibilitou uma análise mais precisa acerca dos temas que despertaram maior interesse do público ao longo do período monitorado. O objetivo consistiu em verificar se conteúdos como processos seletivos, editais ou campanhas institucionais obtiveram maior número de visualizações em comparação a publicações de caráter interno, como comunicados administrativos. Essa abordagem permitiu identificar padrões de engajamento associados ao tipo de conteúdo, fornecendo subsídios mais concretos para compreender quais temáticas efetivamente atraem maior atenção dos leitores.

O Código 12 apresenta o procedimento de agrupamento das notícias a partir das colunas `assunto` e `data_coleta`, com o objetivo de calcular o total de acessos correspondente a cada combinação dessas variáveis. Para isso, foi executada inicialmente uma consulta SQL diretamente no banco SQLite, restringindo a seleção apenas aos campos necessários (`assunto`, `acessos` e `data_coleta`). Essa estratégia contribuiu para a redução do volume de dados transferidos e para a otimização do desempenho da operação.

```python
import sqlite3
import pandas as pd
import numpy as np

# Conectando ao banco de dados SQLite
with sqlite3.connect('database.db') as conn:
    # Consulta SQL para recuperar os campos necessários
    query = 'SELECT assunto, acessos, data_coleta FROM noticia;'
    df = pd.read_sql_query(query, conn)

# Agrupando os dados por assunto e data
grupos = ['assunto', 'data_coleta']
df_agrupado = df.groupby(grupos)\
    .apply(lambda x: np.sum(x['acessos']))\
    .reset_index()

# Renomeando a coluna gerada pela agregação
df_agrupado.rename(columns={0: 'acessos'}, inplace=True)
```

Código 12: Agrupamento das notícias por assunto e data de coleta

Na linha 13, os dados recuperados são organizados utilizando o método `groupby` da biblioteca `pandas`, possibilitando a agregação da quantidade de acessos por meio da função `numpy.sum`. Esse procedimento gera um novo *data frame*, no qual cada registro corresponde ao total de visualizações de um determinado assunto em um dia específico dentro do período de coleta. Posteriormente, a coluna resultante dessa agregação é renomeada para `acessos`, conferindo maior clareza e legibilidade ao conjunto de dados final.

```
1  # Lista de todos os assuntos únicos presentes no dataframe
2  assuntos = df_agrupado['assunto'].unique().tolist()
3
4  for assunto in assuntos:
5      # Filtra os dados apenas para o assunto atual
6      df_local = df_agrupado.loc[df_agrupado['assunto'] == assunto].copy()
7
8      # Converte a coluna de data para o formato datetime
9      df_local['data_coleta'] = pd.to_datetime(df_local['data_coleta'])
10
11     # Cria uma nova coluna com o nome do dia da semana
12     df_local['dia_semana'] = df_local['data_coleta'].apply(
13         lambda data: get_pt_weekday(data.weekday())
14     )
15
16     # Calcula a diferença de acessos em relação ao dia anterior
17     df_local['diff_acessos'] = df_local['acessos'].diff()
```

Código 13: Cálculo da diferença de acessos diários para cada assunto

Dando continuidade à análise da quantidade de acessos por assunto, o Código 13 ilustra o procedimento empregado para calcular a variação diária dos acessos. Inicialmente, é realizada a extração da lista de todos os assuntos únicos presentes no *data frame* previamente agrupado. Esse passo é executado na linha 2 por meio da combinação dos métodos `unique` e `tolist` da biblioteca `pandas`.

Na sequência, a partir da linha 4, itera-se sobre a lista de assuntos únicos obtida na etapa anterior. Para cada assunto, selecionam-se todas as notícias correspondentes, procedimento realizado na linha 6 por meio de uma filtragem utilizando o método `loc` da biblioteca `pandas`, de modo que as operações subsequentes sejam aplicadas exclusivamente aos dados daquele grupo. Em seguida, aplica-se o método `copy` para gerar um novo *data frame*, com o objetivo de prevenir possíveis inconsistências decorrentes de

alterações inadvertidas no *data frame* global.

Com o subconjunto filtrado, na linha 9 a coluna `data_coleta` é convertida para o tipo `datetime`, possibilitando a realização de operações temporais sobre seus valores, como a ordenação cronológica e a extração do dia da semana. A partir dessa conversão, na linha 12 é criada uma nova coluna denominada `dia_semana`, que armazena o nome do dia correspondente a cada data de coleta.

Essa informação adicional revelou-se relevante para análises subsequentes, especialmente na comparação do engajamento entre os diferentes dias úteis da semana. Para esse fim, emprega-se a função `get_pt_weekday`, desenvolvida pelos autores, a qual recebe como entrada um valor inteiro correspondente ao dia da semana e retorna sua representação descritiva em língua portuguesa (por exemplo, 0 para segunda-feira, 1 para terça-feira e assim sucessivamente). A implementação dessa função encontra-se disponível na seção de Apêndices deste trabalho.

Por fim, a coluna `diff_acessos` é gerada na linha 17 por meio da aplicação do método `diff` da biblioteca `pandas`, que calcula, para cada registro, a diferença do número de acessos em relação ao registro anterior no *data frame*. Esse procedimento possibilitou avaliar a variação do interesse do público ao longo do período de coleta, considerando não apenas o total acumulado de acessos, mas também a evolução diária dessa métrica. Tal análise foi fundamental para identificar picos, quedas e padrões de comportamento associados a determinados temas.

A Tabela 2 apresenta os resultados obtidos a partir da execução do código de agregação e cálculo da diferença diária de acessos para o assunto “Ações de Extensão”. Nela, os dados de acessos estão organizados por data de coleta, evidenciando tanto o total acumulado em cada dia quanto a variação no número de acessos em relação ao dia imediatamente anterior.

Esses dados permitem uma análise detalhada do comportamento do público em relação às notícias associadas a esse tema específico. A variação diária de acessos, apresentada na última coluna, evidencia picos e quedas de interesse, permitindo a identificação de períodos de maior engajamento. Adicionalmente, o cruzamento dessa informação com o dia da semana fornece insights sobre padrões semanais de consumo,

Tabela 2: Variação diária da quantidade de acessos para o assunto “Ações de Extensão” durante o período da coleta

Assunto	Data da Coleta	Acessos	Dia da Semana	Diferença de Acessos
acoes de extensao	2023-01-26 21:00:04	865117	quinta	–
acoes de extensao	2023-01-27 21:00:04	865263	sexta	146
acoes de extensao	2023-01-28 21:00:05	865433	sabado	170
acoes de extensao	2023-01-29 21:00:03	865630	domingo	197
acoes de extensao	2023-01-30 21:00:05	865839	segunda	209
acoes de extensao	2023-01-31 21:00:04	866047	terca	208
acoes de extensao	2023-02-01 21:00:05	866278	quarta	231
acoes de extensao	2023-02-02 23:18:22	867775	quinta	1497
acoes de extensao	2023-02-03 21:00:04	868030	sexta	255
acoes de extensao	2023-02-04 21:36:49	869245	sabado	1215
acoes de extensao	2023-02-05 21:00:04	869491	domingo	246
acoes de extensao	2023-02-06 21:00:04	870005	segunda	514
acoes de extensao	2023-02-07 21:04:00	871065	terca	1060
acoes de extensao	2023-02-08 18:00:06	871284	quarta	219
acoes de extensao	2023-02-09 18:06:58	871944	quinta	660
acoes de extensao	2023-02-10 18:00:04	872156	sexta	212
acoes de extensao	2023-02-11 20:38:20	873095	sabado	939
acoes de extensao	2023-02-12 18:17:43	873244	domingo	149
acoes de extensao	2023-02-13 23:22:05	873820	segunda	576

Fonte: Elaborado pelo autor.

os quais podem subsidiar a definição de estratégias mais eficazes para a divulgação e publicação de conteúdos.

6.3.1.2 Pela quantidade de imagens e pela data da coleta

Nesta segunda etapa da análise, buscou-se investigar a existência de uma possível relação entre a quantidade de imagens presentes no corpo das notícias e a variação no número de acessos ao longo do tempo. Para tanto, as notícias foram agrupadas de acordo com o número total de imagens e a data de coleta, com o objetivo de verificar se publicações contendo maior quantidade de elementos gráficos apresentam maior volume de acessos pelos usuários do site. Nessa abordagem, aplicou-se a mesma lógica utilizada anteriormente, agora considerando a presença de imagens como a variável principal de análise.

O Código 14 ilustra o procedimento empregado para agrupar os dados de acessos das notícias com base na quantidade de imagens presentes em seu conteúdo e na data de coleta. Na linha 4, a consulta SQL é executada diretamente no banco SQLite,

utilizando a *query* juntamente com o objeto de conexão e retornando exclusivamente os campos necessários para a análise.

```
1  with sqlite3.connect('database.db') as conn:
2      # Executa a consulta SQL trazendo apenas os campos necessários
3      query = 'SELECT imagens, acessos, data_coleta FROM noticia;'
4      df = pd.read_sql_query(query, conn)
5
6      # Define os campos de agrupamento
7      grupos = ['imagens', 'data_coleta']
8
9      # Agrupa os dados e soma os acessos
10     df_agrupado = df.groupby(grupos) \
11                     .apply(lambda x: np.sum(x['acessos'])) \
12                     .reset_index()
13
14     # Renomeia a coluna de resultado para 'acessos'
15     df_agrupado.rename(columns={0: 'acessos'}, inplace=True)
```

Código 14: Agrupamento das notícias pela quantidade de imagens e a data da coleta

Em seguida, utilizando a biblioteca `pandas`, os dados são agrupados por `imagens` e `data_coleta` na linha 10, sendo a soma total dos acessos calculada por meio da função `apply` em conjunto com `numpy.sum`. A coluna resultante dessa agregação é, então, renomea- da para `acessos`, tornando o *data frame* mais claro e adequado para as etapas subsequentes da análise.

O Código 15 apresenta o cálculo da variação diária de acessos para cada grupo de notícias com igual quantidade de imagens. O procedimento tem início com a criação da coluna `diff_acessos` na linha 2, a qual é preenchida com os valores resultantes da diferença de acessos entre dias consecutivos. Posteriormente, na linha 5, a coluna `data_coleta` é convertida para o tipo `datetime`, assegurando que operações temporais possam ser aplicadas de forma adequada.

```
1  # Inicializa a coluna de diferença
2  df_agrupado['diff_acessos'] = np.nan
3
4  # Converte a coluna de datas para o formato datetime
5  df_agrupado['data_coleta'] = pd.to_datetime(df_agrupado['data_coleta'])
6
7  # Obtém as diferentes quantidades de imagens
8  imagens_distintas = df_agrupado['imagens'].unique().tolist()
9
10 # Para cada grupo de imagens, calcula a variação diária de acessos
11 for imagem in imagens_distintas:
12     grupo_local = df_agrupado.loc[df_agrupado['imagens'] == imagem].copy()
13
14     grupo_local['diff_acessos'] = grupo_local['acessos'].diff()
15
16     df_agrupado.loc[grupo_local.index, 'diff_acessos'] = grupo_local['diff_acessos']
```

Código 15: Cálculo da diferença de acessos por quantidade de imagens

Em seguida, na linha 8, o código identifica todas as quantidades distintas de imagens presentes no *data frame*, utilizando os métodos `unique` e `tolist` da biblioteca *pandas*, e executa um laço *for* com o objetivo de segmentar as notícias em grupos de acordo com cada quantidade de imagens identificada. A cada iteração, selecionam-se as notícias que contêm exatamente aquela quantidade de imagens, sendo criado um novo `data frame` por meio do método `copy`, de forma a gerar um subconjunto de dados isolado.

Dentro de cada subconjunto, aplica-se o mesmo método `diff` da biblioteca `pandas` utilizado anteriormente para calcular a diferença de acessos entre notícias agrupadas por tema. Essa abordagem permite determinar a variação diária de acessos de maneira independente para cada grupo, evitando qualquer interferência entre conjuntos distintos de dados.

Por fim, na linha 16, os resultados calculados são reinseridos no *data frame* original, na coluna `diff_acessos` criada previamente, preenchendo-a com as diferenças de acessos correspondentes aos mesmos índices. Esse procedimento é realizado por meio do atributo `index` da biblioteca *pandas* e é fundamental para possibilitar uma análise detalhada da variação de acessos ao longo do tempo, considerando a quantidade de imagens nas publicações como fator de influência no comportamento dos usuários.

A Tabela 3 apresenta os dados de variação diária da quantidade de acessos referentes exclusivamente às notícias que continham apenas uma imagem em seu conteúdo. As colunas exibem, respectivamente, o número de imagens, a data de coleta, o total acumulado de acessos até aquele dia, a diferença em relação ao dia anterior e o dia da semana correspondente. Essa segmentação por grupo visual possibilitou isolar o comportamento específico de um subconjunto frequente entre as publicações, facilitando a identificação de padrões sutis que poderiam passar despercebidos em uma análise global.

Tabela 3: Variação diária da quantidade de acessos para notícias com apenas uma imagem ao longo do período de coleta

Imagens	Data da Coleta	Acessos	Diferença de Acessos	Dia da Semana
1	2023-01-26 21:00:04	2749649	–	quinta
1	2023-01-27 21:00:04	2750024	375	sexta
1	2023-01-28 21:00:05	2750406	382	sábado
1	2023-01-29 21:00:03	2750859	453	domingo
1	2023-01-30 21:00:05	2751360	501	segunda
1	2023-01-31 21:00:04	2751829	469	terça
1	2023-02-01 21:00:05	2752363	534	quarta
1	2023-02-02 23:18:22	2787327	34964	quinta
1	2023-02-03 21:00:04	2788296	969	sexta
1	2023-02-04 21:36:49	2789134	838	sábado
1	2023-02-05 21:00:04	2789927	793	domingo
1	2023-02-06 21:00:04	2791814	1887	segunda
1	2023-02-07 21:04:00	2794997	3183	terça
1	2023-02-08 18:00:06	2795771	774	quarta
1	2023-02-09 18:06:58	2798546	2775	quinta
1	2023-02-10 18:00:04	2801411	2865	sexta
1	2023-02-11 20:38:20	2804277	2866	sábado
1	2023-02-12 18:17:43	2804938	661	domingo
1	2023-02-13 23:22:05	2808894	3956	segunda

Fonte: Elaborado pelo autor.

É possível observar que os acessos apresentaram relativa estabilidade nos primeiros dias, com variações pequenas e regulares. Entretanto, verificam-se momentos em que a diferença diária aumenta abruptamente, como no dia 2 de fevereiro, quando os acessos cresceram mais de 31 mil em relação ao dia anterior.

Esse comportamento pode indicar a ocorrência de eventos pontuais que atraíram maior atenção do público, mesmo dentro de uma categoria de notícias visualmente mais simples. A análise dessas flutuações, considerando o número de imagens como critério de segmentação, contribui para uma compreensão mais detalhada da dinâmica de engajamento dos leitores em relação aos diferentes formatos de conteúdo.

6.3.1.3 Pela presença ou ausência de imagem de perfil e pela data da coleta

Na terceira etapa da análise, adotou-se a mesma abordagem utilizada nas consultas anteriores, porém considerando um novo critério: a presença ou ausência de uma imagem de capa nas notícias. Essa informação é registrada na coluna `tem_imagem_perfil`, a qual indica se determinada publicação contém esse tipo de destaque visual. O objetivo dessa etapa foi investigar se a presença da imagem de capa, frequentemente utilizada para conferir maior apelo visual à notícia, exerce influência sobre o comportamento de acesso dos leitores. Um exemplo ilustrativo dessa distinção é apresentado na Figura 13.

Figura 13: Notícia em destaque com imagem de perfil marcada em vermelho.

Fonte: Elaborado pelo autor.

A consulta ao banco de dados foi realizada considerando o atributo `tem_imagem_perfil` em conjunto com a data de coleta, possibilitando a análise da variação de acessos ao longo do tempo para notícias com e sem imagem de capa. O agrupamento dos dados e o cálculo da soma de acessos são detalhados no Código 16, enquanto o Código 17

apresenta a lógica empregada para o cálculo das diferenças diárias. Os resultados obtidos encontram-se nas Tabelas 4 e 5, permitindo uma comparação direta entre os dois cenários analisados.

O Código 16 ilustra o procedimento de agregação dos dados de acesso com base na presença ou ausência de imagem de capa nas notícias. A consulta SQL, realizada diretamente no banco de dados, retorna apenas os campos necessários, conforme demonstrado nas linhas 3 e 4, tornando o processo mais eficiente.

```
1  with sqlite3.connect('database.db') as conn:
2      # Executa a consulta SQL para trazer os campos necessários
3      query = 'SELECT tem_imagem_perfil, acessos, data_coleta FROM noticia;'
4      df = pd.read_sql_query(query, conn)
5
6      # Define os campos utilizados no agrupamento
7      grupos = ['tem_imagem_perfil', 'data_coleta']
8
9      # Realiza o agrupamento e soma os acessos para cada combinação
10     df_agrupado = df.groupby(grupos) \
11                     .apply(lambda x: np.sum(x['acessos'])) \
12                     .reset_index()
13
14     # Renomeia a coluna de resultado para 'acessos'
15     df_agrupado.rename(columns={0: 'acessos'}, inplace=True)
```

Código 16: Agrupamento das notícias pela presença de imagem de perfil e data da coleta

Em seguida, na linha 10, os dados são agrupados pelas colunas `tem_imagem_perfil` e `data_coleta`, somando-se os acessos correspondentes a cada combinação. O resultado é um *data frame* que apresenta, dia a dia, o total de visualizações das publicações separadas por esse critério visual, permitindo a comparação entre os dois grupos ao longo do período analisado.

O Código 17 apresenta a lógica empregada para calcular a variação diária de acessos das notícias, considerando a presença ou ausência de imagem de capa. O procedimento inicia-se com a criação da coluna `diff_acessos` na linha 2, destinada a armazenar os valores de diferença entre os totais de acessos registrados em dias consecutivos.

```
1  # Cria coluna para armazenar a diferença de acessos
2  df_agrupado['diff_acessos'] = np.nan
3
4  # Converte a coluna de datas para datetime
5  df_agrupado['data_coleta'] = pd.to_datetime(df_agrupado['data_coleta'])
6
7  # Obtém os valores distintos da coluna 'tem_imagem_perfil'
8  grupos = df_agrupado['tem_imagem_perfil'].unique().tolist()
9
10 # Calcula a diferença de acessos para cada grupo
11 for grupo in grupos:
12     df_local = df_agrupado.loc[df_agrupado['tem_imagem_perfil'] == grupo].copy()
13
14     # Gera o nome do dia da semana em português
15     df_local['dia_semana'] = df_local['data_coleta'] \
16         .apply(lambda dt: get_pt_weekday(dt.weekday()))
17
18     # Calcula a variação de acessos dia a dia
19     df_local['diff_acessos'] = df_local['acessos'].diff()
20
21     df_agrupado.loc[df_local.index, ['diff_acessos', 'dia_semana']] = \
22         df_local[['diff_acessos', 'dia_semana']]
```

Código 17: Cálculo da variação de acessos por presença de imagem de perfil

Na linha 5, a coluna `data_coleta` é convertida para o tipo `datetime`, permitindo a execução de operações temporais com maior precisão, como a extração do dia da semana. Em seguida, nas linhas 8 e 11, os valores únicos da coluna `tem_imagem_perfil` são identificados por meio dos métodos `unique` e `tolist` da biblioteca *pandas*, correspondendo às duas categorias analisadas, e cada valor é processado individualmente através de um laço `for`.

Dentro do laço `for`, o *data frame* é filtrado de acordo com cada grupo distinto, sendo criado um novo *data frame* por cópia, conforme realizado nas análises anteriores. Em seguida, é criada a coluna `dia_semana`, preenchida a partir da coluna de data de coleta e utilizando a função auxiliar `get_pt_weekday` para retornar os nomes descritivos dos dias da semana em língua portuguesa.

A diferença de acessos é calculada utilizando o método `diff` na linha 19, que subtrai os valores consecutivos da coluna `acessos`, possibilitando a avaliação da variação

do interesse diário dos leitores. Por fim, os resultados são reinseridos no *data frame* original na linha 21, consolidando as novas colunas com as informações necessárias para análises quantitativas subsequentes.

A Tabela 4 apresenta os dados consolidados das notícias que não possuem imagem de perfil, organizados por data de coleta. Cada linha corresponde a um dia específico, contendo o total acumulado de acessos registrado até aquele momento. A coluna `diff_acessos` indica a diferença no número de acessos em relação ao dia anterior, permitindo observar a variação diária.

Tabela 4: Variação diária da quantidade de acessos para notícias sem imagem de perfil durante o período de coleta

Imagem de Perfil	Data da Coleta	Acessos	Diferença de Acessos	Dia da Semana
não	2023-01-26 21:00:04	15683	–	quinta
não	2023-01-27 21:00:04	15685	2	sexta
não	2023-01-28 21:00:05	15685	0	sábado
não	2023-01-29 21:00:03	15686	1	domingo
não	2023-01-30 21:00:05	15691	5	segunda
não	2023-01-31 21:00:04	15691	0	terça
não	2023-02-01 21:00:05	15692	1	quarta
não	2023-02-02 23:18:22	15694	2	quinta
não	2023-02-03 21:00:04	15698	4	sexta
não	2023-02-04 21:36:49	15699	1	sábado
não	2023-02-05 21:00:04	15703	4	domingo
não	2023-02-06 21:00:04	15706	3	segunda
não	2023-02-07 21:04:00	15720	14	terça
não	2023-02-08 18:00:06	15722	2	quarta
não	2023-02-09 18:06:58	15731	9	quinta
não	2023-02-10 18:00:04	15731	0	sexta
não	2023-02-11 20:38:20	15736	5	sábado
não	2023-02-12 18:17:43	15736	0	domingo
não	2023-02-13 23:22:05	15741	5	segunda

Fonte: Elaborado pelo autor.

A coluna `dia_semana`, gerada automaticamente a partir da data de coleta, indica o respectivo dia da semana em que os registros foram realizados. Essa estrutura possibilita a análise do comportamento dos acessos ao longo do tempo, segmentando as publicações de acordo com sua característica visual.

De forma semelhante, a Tabela 5 apresenta o quantitativo diário de acessos das notícias que contêm imagem de perfil durante o período monitorado. Observa-se que os acessos neste grupo foram significativamente maiores em comparação ao grupo sem imagem de perfil. No conjunto das notícias sem imagem de perfil, houve dias em que a diferença diária de acessos permaneceu zerada, indicando ausência de variação, como

Tabela 5: Variação diária da quantidade de acessos para notícias com imagem de perfil durante o período de coleta

Imagem de Perfil	Data da Coleta	Acessos	Diferença de Acessos	Dia da Semana
sim	2023-01-26 21:00:04	3330247	–	quinta
sim	2023-01-27 21:00:04	3330720	473	sexta
sim	2023-01-28 21:00:05	3331205	485	sábado
sim	2023-01-29 21:00:03	3331796	591	domingo
sim	2023-01-30 21:00:05	3332422	626	segunda
sim	2023-01-31 21:00:04	3333010	588	terça
sim	2023-02-01 21:00:05	3333658	648	quarta
sim	2023-02-02 23:18:22	3369033	35375	quinta
sim	2023-02-03 21:00:04	3370165	1132	sexta
sim	2023-02-04 21:36:49	3372133	1968	sábado
sim	2023-02-05 21:00:04	3373073	940	domingo
sim	2023-02-06 21:00:04	3375218	2145	segunda
sim	2023-02-07 21:04:00	3378981	3763	terça
sim	2023-02-08 18:00:06	3379904	923	quarta
sim	2023-02-09 18:06:58	3383033	3129	quinta
sim	2023-02-10 18:00:04	3386021	2988	sexta
sim	2023-02-11 20:38:20	3389369	3348	sábado
sim	2023-02-12 18:17:43	3390116	747	domingo
sim	2023-02-13 23:22:05	3394364	4248	segunda

Fonte: Elaborado pelo autor.

nos dias 28/01, 31/01, 10/02 e 12/02 (sendo que a maioria desses dias corresponde a fins de semana, exceto 31/01, que foi uma terça-feira).

Entretanto, nas notícias com imagem de perfil, não foi observado nenhum dia sem registro de acessos; pelo contrário, verificou-se certa regularidade, bem como um aumento gradual da métrica ao longo do tempo. Dessa forma, pode-se inferir que a presença de uma imagem de perfil constitui um fator relevante para o incremento de acessos por parte dos usuários do site do IF Goiano.

6.3.1.4 Pelo dia da semana da coleta

Na análise subsequente, buscou-se avaliar a quantidade de acessos das notícias em função do dia da semana, com o objetivo de identificar aqueles que apresentaram maior volume total de acessos. Para isso, inicialmente calculou-se o total de acessos por data de coleta utilizando o método `groupby` da biblioteca `pandas` em conjunto com a função `sum` do `numpy`. O Código 18 ilustra detalhadamente o procedimento empregado, que segue lógica similar às análises previamente descritas.

Conforme demonstrado no Código 18, o procedimento adota abordagem semelhante à das análises anteriores. Inicialmente, nas linhas 1 a 4, realiza-se a leitura das colunas `acessos` e `data_coleta`, extraindo do banco de dados SQLite apenas os dados necessários para a análise. Em seguida, as notícias são agrupadas por data de coleta, e realiza-se a soma dos acessos nas linhas 7 a 10, finalizando com a padronização do nome da coluna contendo o total de acessos para `acessos`.

```
1  with sqlite3.connect('database.db') as conn:
2      # Executa a consulta SQL para trazer os campos necessários
3      query = 'SELECT acessos, data_coleta FROM noticia;'
4      df = pd.read_sql_query(query, conn)
5
6      # Define os campos utilizados no agrupamento
7      grupos = ['data_coleta']
8
9      # Realiza o agrupamento e soma os acessos para cada data
10     df_agrupado = df.groupby(grupos) \
11                     .apply(lambda x: np.sum(x['acessos'])) \
12                     .reset_index()
13
14     # Renomeia a coluna de resultado para 'acessos'
15     df_agrupado.rename(columns={0: 'acessos'}, inplace=True)
```

Código 18: Cálculo da quantidade total de acessos das notícias agrupadas pela data da coleta.

Após a soma do quantitativo total de acessos por data de coleta, o passo seguinte consiste na aplicação do método `diff` da biblioteca `pandas`, a fim de calcular a variação diária de acessos das notícias, conforme ilustrado no Código 19, na linha 2. Paralelamente, a coluna `data_coleta` é convertida para o tipo `datetime` na linha 3, o que possibilita a utilização do método `weekday` para extrair o dia da semana correspondente. Esse valor será posteriormente empregado na etapa de agregação, permitindo calcular o total de acessos distribuídos por dia da semana no código subsequente.

```
1  # Calcula a diferença de acessos entre datas de coleta consecutivas
2  df_agrupado['acessos'] = df_agrupado['acessos'].diff()
3  df_agrupado['data_coleta'] = pd.to_datetime(df_agrupado['data_coleta'])
```

Código 19: Cálculo da diferença de acessos das notícias agrupadas pela data da coleta.

O Código 20 apresenta a etapa final do procedimento para obtenção do quantitativo de acessos das notícias segmentados por dia da semana. Na linha 2, realiza-se a extração do dia da semana a partir do método `weekday` do tipo `datetime`, seguido da aplicação da função auxiliar `get_pt_weekday`, responsável por converter o valor numérico em sua representação textual em língua portuguesa.

```
1  # Extrai o dia da semana em português
2  df_agrupado['dia_semana'] = df_agrupado['data_coleta'].apply(
3      lambda data_coleta: get_pt_weekday(data_coleta.weekday())
4  )
5
6  # Agrupa e soma os acessos por dia da semana
7  df_dia_semana = df_agrupado[['dia_semana', 'acessos']].groupby('dia_semana') \
8                      .sum(numeric_only=True) \
9                      .reset_index()
10
11 # Converte para inteiro
12 df_dia_semana['acessos'] = df_dia_semana['acessos'].astype(int)
13
14 # Define a ordem dos dias da semana
15 ordem_dias_semana = ['domingo', 'segunda', 'terca', 'quarta',
16                       'quinta', 'sexta', 'sabado']
17
18 df_dia_semana['dia_semana'] = pd.Categorical(df_dia_semana['dia_semana'],
19                                               categories=ordem_dias_semana,
20                                               ordered=True)
21
22 # Ordena pelos dias da semana
23 df_dia_semana.sort_values(by='dia_semana', inplace=True)
```

Código 20: Cálculo do total de acessos das notícias pelo dia da semana.

Em seguida, selecionam-se apenas as colunas `dia_semana` e `acessos`, que servem de base para o agrupamento dos registros conforme o dia da semana e para o cálculo da soma total de acessos associados a cada grupo, conforme ilustrado nas linhas 7 a 9. Após essa etapa, nas linhas 15 a 20, procede-se à ordenação dos dados utilizando o tipo `Categorical`, que permite definir de forma parametrizável a ordem natural dos elementos, em conjunto com o método `sort_values`, ambos disponibilizados pela biblioteca *pandas*. Nesse caso, a ordenação foi realizada segundo os valores descritivos contidos na coluna `dia_semana`.

O resultado do procedimento descrito é apresentado na Tabela 6, que exibe o quantitativo obtido a partir da soma das diferenças de acessos diários, agrupados por dia da semana. Em outras palavras, a tabela reflete o total de acessos consolidados para cada dia da semana.

Tabela 6: Soma total de acessos de notícias por dia da semana

Dia da Semana	Acessos
domingo	2283
segunda	7032
terça	4365
quarta	1574
quinta	38515
sexta	4599
sábado	5807

Fonte: Elaborado pelo autor.

Para calcular a média de acessos, a fim de obter uma visualização mais equilibrada e representativa do comportamento dos usuários ao longo da semana, basta alterar a linha 8 do Código 20, substituindo o método `sum` pelo método `mean`, ambos pertencentes à biblioteca *pandas*. O resultado desse cálculo é apresentado na Tabela 7.

Tabela 7: Média de acessos de notícias por dia da semana

Dia da Semana	Acessos
domingo	761
segunda	2344
terça	2182
quarta	787
quinta	19257
sexta	1533
sábado	1935

Fonte: Elaborado pelo autor.

6.3.1.5 Pelo dia da semana da coleta e pelo assunto

Por fim, na última consulta, as notícias foram agrupadas simultaneamente pelo dia da semana e pelo respectivo assunto, calculando-se a diferença de acessos entre dias consecutivos. O objetivo foi identificar os padrões de engajamento associados a

cada tema ao longo da semana, possibilitando verificar em quais dias determinados assuntos alcançaram maior número de acessos. A implementação encontra-se detalhada nos Códigos 21 e 22.

```
1  with sqlite3.connect('database.db') as conn:
2      # Executa a consulta SQL para trazer os campos necessários
3      query = 'SELECT acessos, data_coleta, assunto FROM noticia;'
4      df = pd.read_sql_query(query, conn)
5
6      # Define os campos utilizados no agrupamento
7      grupos = ['data_coleta', 'assunto']
8
9      # Realiza o agrupamento e soma os acessos para cada combinação
10     df_agrupado = df.groupby(grupos) \
11                     .sum(numeric_only=True) \
12                     .reset_index()
13
14     # Extrai o dia da semana em português
15     df_agrupado['dia_semana'] = df_agrupado['data_coleta'].apply(
16         lambda data_coleta: get_pt_weekday(pd.to_datetime(data_coleta).weekday())
17     )
18
19     # Define a ordem dos dias da semana
20     ordem_dias_semana = ['domingo', 'segunda', 'terca', 'quarta',
21                          'quinta', 'sexta', 'sabado']
22
23     # Converte para categoria ordenada
24     df_agrupado['dia_semana'] = pd.Categorical(df_agrupado['dia_semana'],
25                                                categories=ordem_dias_semana,
26                                                ordered=True)
```

Código 21: Cálculo do total de acessos das notícias pela data de coleta e pelo assunto.

Assim como nas etapas anteriores, nas linhas 1 a 4 são selecionadas apenas as colunas necessárias do banco de dados, sendo elas `acessos`, `data_coleta` e `assunto`. Em seguida, entre as linhas 10 e 12, os dados são agrupados por `data_coleta` e `assunto`, obtendo-se a soma total de acessos para cada combinação. Posteriormente, entre as linhas 15 e 17, realiza-se a extração do dia da semana correspondente a cada data de coleta e, por fim, entre as linhas 20 e 26, define-se sua ordenação natural por meio do tipo `Categorical` da biblioteca *pandas*.

```
1  # Obtém a lista única de assuntos
2  assuntos = df['assunto'].unique().tolist()
3
4  for assunto in assuntos:
5      df_assunto = df_agrupado[df_agrupado['assunto'] == assunto].copy()
6      df_assunto['acessos'] = df_assunto['acessos'].diff()
7
8      df_semanal = df_assunto[['acessos', 'assunto', 'dia_semana']].copy() \
9          .groupby(['dia_semana', 'assunto'], observed=True) \
10         .sum(numeric_only=True) \
11         .reset_index()
12
13     df_semanal.sort_values(by='dia_semana', inplace=True)
```

Código 22: Cálculo do total de acessos das notícias pelo dia da semana e pelo assunto.

Para finalizar essa análise, o Código 22 mostra, nas linhas 2 a 4, a obtenção dos assuntos únicos para posterior agrupamento e a construção de um laço `for` que percorre cada um deles. Em seguida, na linha 5, realiza-se o filtro das linhas correspondentes ao assunto em iteração, gerando um `dataframe` individual, em processo semelhante ao utilizado nas análises anteriores.

Logo, na linha 6, calcula-se a diferença de acessos das notícias do assunto filtrado entre as datas de coleta, utilizando o método `diff`, etapa essencial para viabilizar a análise. Em seguida, entre as linhas 8 e 11, realiza-se o último agrupamento pelas colunas `dia_semana` (gerada no código anterior) e `assunto`, obtendo-se a soma total de acessos. Por fim, na linha 13, procede-se à ordenação dos valores conforme o dia da semana, recorrendo à ordenação natural previamente definida com `Categorical`.

Assim, como conclusão do código desenvolvido para as análises, observa-se uma linearidade e padronização na metodologia adotada. Em todos os agrupamentos e no cálculo da métrica de acessos, manteve-se a mesma lógica das consultas anteriores, com o uso das bibliotecas `pandas` e `numpy`. As diferenças limitaram-se às colunas consideradas em cada conjunto de dados. Como resultado, obteve-se um panorama abrangente do quantitativo de acessos por assunto em cada dia da semana, conforme apresentado nas Tabelas 8 a 12.

Tabela 8: Quantidade de acessos das notícias com tema ”processos seletivos”por dia da semana

Dia da Semana	Variação de Acessos
domingo	692
segunda	3017
terça	1076
quarta	404
quinta	11858
sexta	2055
sábado	1364

Fonte: Elaborado pelo autor.

Tabela 9: Quantidade de acessos das notícias com tema ”editais”por dia da semana

Dia da Semana	Variação de Acessos
domingo	694
segunda	1868
terça	1345
quarta	457
quinta	19435
sexta	1520
sábado	1424

Fonte: Elaborado pelo autor.

Tabela 10: Quantidade de acessos das notícias com tema ”ações de extensão”por dia da semana.

Dia da Semana	Variação de Acessos
domingo	592
segunda	1299
terça	1268
quarta	450
quinta	2157
sexta	613
sábado	2324

Fonte: Elaborado pelo autor.

Tabela 11: Quantidade de acessos das notícias com tema ”comunicados oficiais”por dia da semana.

Dia da Semana	Variação de Acessos
domingo	158
segunda	463
terça	332
quarta	139
quinta	4823
sexta	261
sábado	378

Fonte: Elaborado pelo autor.

Tabela 12: Quantidade de acessos das notícias com tema ”campanhas”por dia da semana.

Dia da Semana	Variação de Acessos
domingo	147
segunda	385
terça	344
quarta	124
quinta	242
sexta	150
sábado	317

Fonte: Elaborado pelo autor.

6.3.2 Análise de sentimento

As consultas realizadas para a análise da quantidade de acessos foram desenvolvidas a partir dos dados coletados das notícias, com o objetivo de identificar correlações entre as características das publicações e a sua popularidade. Para isso, empregaram-se as bibliotecas para análise de dados `pandas` e `numpy` da linguagem Python, implementando algoritmos básicos de Análise Exploratória de Dados (da sigla em inglês EDA, *Exploratory Data Analysis*).

A etapa seguinte consiste em analisar o sentimento presente nos títulos e descrições das notícias, correlacionando-o com a quantidade de acessos obtida. Busca-se verificar se conteúdos com tom positivo atraem maior número de acessos ou se, ao contrário, os negativos despertam maior engajamento. Considera-se também a possibilidade de o tom neutro se destacar, uma vez que as notícias institucionais, em geral, apresentam caráter informativo e tendem a evitar posicionamentos opinativos.

Para a análise de sentimento dos textos das notícias, empregou-se o modelo pré-treinado *Bidirectional Encoder Representations from Transformers* (BERT), conforme descrito por Devlin *et al.* (2019). Neste trabalho, optou-se pela utilização da implementação disponibilizada pela biblioteca *Transformers*, também da linguagem Python, a qual possibilita o uso do modelo de forma prática e eficiente, oferecendo um método simples para a indicação precisa do sentimento presente no material textual analisado.

A biblioteca *Transformers* é desenvolvida e mantida pela empresa norte-americana Hugging Face[10], a qual disponibiliza uma plataforma colaborativa voltada a desenvolvedores e pesquisadores da área de *machine learning* (aprendizado de máquina) para o

[10]O endereço oficial para acesso se encontra em: https://huggingface.co/

compartilhamento de modelos, *datasets* e aplicações de inteligência artificial. A *Transformers* provê milhares de outros modelos pré-treinados capazes de realizar tarefas de Processamento de Linguagem Natural (da sigla em inglês NLP, *Natural Language Processing*) em diferentes modalidades, como texto, visão e áudio.

Para utilizar um dos diversos modelos disponibilizados, cria-se um objeto `pipeline` e informa-se o nome do modelo desejado, conforme ilustrado nas linhas 4 e 5 do Código 23. Neste exemplo, um objeto `pipeline` é instanciado na linha 4, configurado para empregar o modelo `bert-base-multilingual-uncased-sentiment`, de autoria de *nlptown*, para a análise de sentimento de dois textos em língua portuguesa.

```
1  from transformers import pipeline
2
3  # Inicializa o pipeline de análise de sentimentos
4  sentiment = pipeline('sentiment-analysis',
5                       model='nlptown/bert-base-multilingual-uncased-sentiment')
6
7  # Define as frases para análise
8  phrase_1 = 'Estou extremamente feliz e satisfeito com o resultado excelente!'
9  phrase_2 = 'Estou muito triste e desapontado com essa situação terrível.'
10
11 # Executa a análise de sentimentos
12 print(sentiment(phrase_1))
13 print(sentiment(phrase_2))
```

Código 23: Utilizando a biblioteca Transformers para fazer a análise de sentimento de duas frases simples.

Em seguida, para aplicar o modelo aos textos nos quais se deseja identificar o sentimento, basta utilizar a variável que armazenou o objeto `pipeline` como se fosse uma função convencional do Python. Ao invocar `sentiment` sobre `phrase_1` e `phrase_2`, obtém-se, por exemplo, os seguintes resultados (que podem variar conforme a implementação do modelo):

- `phrase_1`: [’label’: ’5 stars’, ’score’: 0.7405913472175598];
- `phrase_2`: [’label’: ’1 star’, ’score’: 0.7489633560180664].

Nesta análise, o campo `label` indica a classificação do sentimento em uma escala de 1 a 5 estrelas, sendo 1 correspondente a muito negativo e 5 a muito positivo. O

campo `score` representa o grau de confiança do modelo na classificação atribuída. Assim, neste exemplo, a frase positiva recebeu 5 estrelas e a frase negativa, 1 estrela, o que é coerente com a expectativa. Além disso, o modelo apresentou alta confiança em suas respostas, com aproximadamente 75% de assertividade.

Desta forma, a partir da plataforma da Hugging Face escolheu-se o modelo `bert-base-multilingual-uncased` do autor NLP Town[11] para avaliar o sentimento das notícias em 2 (dois) momentos principais:

- Título e descrição: dar a cada notícia, uma avaliação de sentimento BERT do seu título e descrição, com uma quantidade de estrelas de 1 (um) a 5 (cinco) e uma pontuação de 0 (zero) a 1 (um), ambos indicando a positividade do texto;
- Descrição do jornalista e descrição gerada pelo BART: avaliar, separadamente, a descrição da notícia e o corpo da notícia sumarizado pelo modelo BART (ver seção 6.3.3), no intuito de verificar qual das descrições obtém, em média, a melhor análise de sentimento: o texto descritivo escrito pela equipe de jornalismo ou a descrição (corpo da notícia resumido) gerada pelo modelo para representação de linguagem BART.

Esse modelo foi ajustado (*fine-tuned*) para análise de sentimento em avaliação de produtos em 6 (seis) idiomas: inglês, holandês, alemão, francês, espanhol e italiano, prevendo o sentimento da avaliação como um número de estrelas entre 1 (um) e 5 (cinco).

A partir do Código 24, observa-se, na linha 6, a criação de um objeto `tokenizer`, responsável pela tokenização (isto é, a segmentação em unidades básicas de texto) dos pares título/descrição de cada notícia, considerando que alguns textos ultrapassam o limite suportado pelo objeto `pipeline`. Já na linha 11, é instanciado um objeto `pipeline`, anteriormente descrito neste trabalho, o qual disponibiliza o algoritmo pré-treinado do *BERT*, ajustado de acordo com o modelo *bert-base-multilingual-uncased-sentiment*.

[11] O endereço oficial para acesso se encontra em: `https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment`. Acesso feito em 14/06/2024.

```
import pandas as pd
import os
from transformers import AutoTokenizer, pipeline

# Carrega o tokenizador do modelo BERT para análise de sentimentos
tokenizador = AutoTokenizer.from_pretrained(
    pretrained_model_name_or_path='nlptown/bert-base-multilingual-uncased-sentiment'
)

# Inicializa o pipeline de análise de sentimentos
analisador_sentimentos = pipeline(
    task='sentiment-analysis',
    model='nlptown/bert-base-multilingual-uncased-sentiment',
    tokenizer=tokenizador  # Usa o tokenizador carregado
)
```

Código 24: Criação dos objetos `tokenizer` e `pipeline` para realizar a análise de sentimento dos textos das notícias.

No Código 25, observa-se, na linha 9, a obtenção dos *tokens* de cada combinação título/descrição de notícia por meio da chamada ao objeto `tokenizer`. Já na linha 18, esses *tokens* são decodificados pelo método `decode` do mesmo objeto e, em seguida, submetidos ao objeto `pipeline` (i.e., `analisador_sentimentos`) configurado para a análise de sentimento. Ao final do processamento, a variável `resultado` armazena a saída da análise no mesmo formato previamente ilustrado em `phrase_1` e `phrase_2`.

Após a análise de sentimento dos pares título/descrição das notícias, iniciou-se a segunda etapa da análise. O procedimento adotado foi semelhante ao anterior, diferenciando-se apenas pelo foco exclusivo na descrição de cada notícia, em substituição aos pares título/descrição, e pela utilização de uma nova descrição gerada pelo modelo *BART*, cuja abordagem será detalhada no próximo tópico, a partir do corpo das notícias coletado por meio de um segundo robô de software desenvolvido especificamente para esse fim.

Dessa forma, tornou-se possível comparar as médias dos valores obtidos na análise de sentimento entre as duas categorias de descrição (jornalistas e *BART*) e identificar qual delas apresentou os melhores índices.

```
# Lista para armazenar os resultados da análise de sentimentos
resultados = []
# Itera explicitamente sobre cada linha do DataFrame
for indice, linha in df.iterrows():
    # Combina título e descrição para análise, separando por ponto
    texto_analise = f"{linha['titulo']}. {linha['descricao']}"
    # Tokeniza o texto para o modelo BERT
    tokens = tokenizador(
        texto_analise,           # Texto a ser tokenizado
        max_length=512,          # Número máximo de tokens por texto
        padding=True,            # Preenche textos menores que max_length
        truncation=True,         # Trunca textos maiores que max_length
        return_tensors="pt"      # Retorna tensores do PyTorch
    )
    # Realiza a análise de sentimentos
    resultado = analisador_sentimentos(
        tokenizador.decode(
            skip_special_tokens=True                # Remove tokens especiais [CLS], [SEP]
        )
    )
    # Armazena os resultados na lista
    resultados.append(resultado[0])
# Adiciona as colunas com os resultados ao DataFrame
df['estrelas_bert'] = [int(resultado['label'].split(' ')[0]) for resultado in resultados]   #
    Extrai o número de estrelas (1-5)
df['pontuacao_bert'] = [resultado['score'] for resultado in resultados]                     #
    Extrai a pontuação de confiança (0-1)
```

Código 25: Análise de sentimento de cada par título/descrição das notícias.

No Código 26, apresenta-se parte do robô de software desenvolvido para coletar exclusivamente o corpo de cada notícia, com o objetivo de submetê-lo à sumarização por meio do modelo *BART*, a fim de gerar uma nova descrição das notícias.

Nas linhas 4 a 10, verifica-se a checagem quanto à existência de uma descrição ou, alternativamente, a extração do subtítulo, de modo a remover posteriormente o texto redundante, caso este também esteja presente no corpo da notícia. Já nas linhas 18, 22 e 24, realiza-se a extração do conteúdo de cada elemento HTML do tipo parágrafo, armazenando-o na variável `corpo_texto`, que representa o texto completo do corpo da notícia.

```python
def _extrair_dados_publicacao(self):
    try:
        # Tenta extrair a DESCRIÇÃO da notícia
        descricao = WebDriverWait(self._driver, self.TEMPO_ESPERA_PADRAO).until(
            ec.presence_of_element_located((By.CLASS_NAME, 'description'))
        ).text.strip().replace('\n', ' ').replace('  ', ' ')
    except (exceptions.NoSuchElementException, exceptions.TimeoutException):
        # Se não encontrar descrição, tenta extrair o SUBTÍTULO
        try:
            descricao = WebDriverWait(self._driver, self.TEMPO_ESPERA_PADRAO).until(
                ec.presence_of_element_located((By.CLASS_NAME, 'subtitle'))
            ).text.strip().replace('\n', ' ').replace('  ', ' ')
        except (exceptions.NoSuchElementException, exceptions.TimeoutException):
            # Se não encontrar nem descrição nem subtítulo, define como None
            descricao = None
    try:
        # Localiza o painel principal de CONTEÚDO da notícia
        painel_conteudo = WebDriverWait(self._driver, self.TEMPO_ESPERA_PADRAO).until(
            ec.presence_of_element_located((By.ID, 'content'))
        )
        # Extrai todos os parágrafos do conteúdo
        paragrafos = painel_conteudo.find_elements(By.TAG_NAME, 'p')
        # Combina o texto de todos os parágrafos em um único corpo
        corpo_texto = ''.join([paragrafo.text for paragrafo in paragrafos])
        corpo_texto = corpo_texto.strip().replace('\n', ' ').replace('  ', ' ')
    except (exceptions.NoSuchElementException, exceptions.TimeoutException):
        # Se não conseguir extrair o conteúdo principal
        corpo_texto = None
    return descricao, corpo_texto
```

Código 26: Robô de software para fazer a coleta do corpo das notícias no site do IF Goiano.

Após a coleta, procedeu-se à sumarização do corpo das notícias utilizando o modelo *BART*, que gerou um resumo para cada uma delas, podendo ser utilizado como nova descrição. O código correspondente é apresentado na próxima seção, a fim de que, nesta, mantenha-se o foco exclusivamente na análise de sentimento das notícias.

Uma vez geradas as novas descrições, resultantes da sumarização do corpo das notícias, prosseguiu-se com a aplicação de uma nova análise de sentimento utilizando o modelo *bert-base-multilingual-uncased-sentiment*. Em seguida, os valores obtidos sobre os resumos produzidos pelo modelo de IA foram comparados às descrições originais redigidas pelos jornalistas, por meio do cálculo das médias, conforme apresentado no Có-

digo 27.

```
1  import pandas as pd
2  from matplotlib import pyplot as plt
3  import numpy as np
4
5  if __name__ == '__main__':
6      # Carrega o dataset com as análises de sentimentos
7      df = pd.read_csv('datasets/pubs_data_sentiment_2024_03_21_18_05_15_1_19_42.csv')
8      # Calcula as médias das estrelas BERT
9      media_estrelas_descricao = round(np.mean(df['descricao_estrelas'].values.tolist()))
10     media_estrelas_corpo_sumarizado = round(np.mean(df['corpo_summ_estrelas'].values.tolist()))
11     # Calcula as médias das pontuações BERT
12     media_pontuacao_descricao = round(np.mean(df['descricao_pontuacao'].values.tolist()), 2)
13     media_pontuacao_corpo_sumarizado = round(np.mean(df['corpo_summ_pontuacao'].values.tolist()),
           2)
14     # Exibe os resultados comparativos
15     print("=== COMPARAÇÃO DAS ANÁLISES DE SENTIMENTO ===")
16     print(f"Média de Estrelas - Descrição (jornalistas): {media_estrelas_descricao}")
17     print(f"Média de Estrelas - Corpo (BART sumarizado): {media_estrelas_corpo_sumarizado}")
18     print()
19     print(f"Média de Pontuação - Descrição (jornalistas): {media_pontuacao_descricao}")
20     print(f"Média de Pontuação - Corpo (BART sumarizado): {media_pontuacao_corpo_sumarizado}")
21     # Calcula as diferenças para análise
22     diferenca_estrelas = media_estrelas_descricao - media_estrelas_corpo_sumarizado
23     diferenca_pontuacao = media_pontuacao_descricao - media_pontuacao_corpo_sumarizado
24     print()
25     print("=== DIFERENÇAS ENTRE AS ANÁLISES ===")
26     print(f"Diferença nas Estrelas: {diferenca_estrelas}")
27     print(f"Diferença na Pontuação: {diferenca_pontuacao}")
```

Código 27: Cálculo da média da quantidade de estrelas e da pontuação obtidas pela análise de sentimento.

As colunas `descricao_estrelas` e `descricao_pontuacao`, presentes no *dataframe*, indicam, respectivamente, a quantidade de estrelas e a pontuação atribuídas pelo modelo *BERT* à descrição elaborada pela equipe de jornalismo (isto é, aquela disponível no site). De forma análoga, as colunas `corpo_summ_estrelas` e `corpo_summ_pontuacao` correspondem aos valores atribuídos à descrição gerada pelo modelo *BART* a partir do corpo das notícias.

No referido código, cada uma das colunas foi acessada individualmente nas linhas 9, 10, 12 e 13, e a média de seus valores foi calculada por meio da função `numpy.mean`,

aplicada sobre os dados convertidos em lista com o uso dos métodos `values` e `tolist` do *dataframe*. Em seguida, realizou-se a comparação entre as médias obtidas na análise de sentimento, a fim de identificar qual tipo de descrição (jornalista ou *BART*) apresentou a maior pontuação.

6.3.3 Sumarização

O processo de sumarização consistiu na aplicação de um modelo de inteligência artificial, neste caso, o *BART*, para gerar novas descrições das notícias a partir do texto presente em seu corpo. O objetivo foi avaliar se o uso de tais modelos poderia produzir descrições com tonalidade emocional mais positiva, considerando a hipótese de que textos com sentimento mais positivo, tendem, em média, a ser mais atrativos, desde que sejam adequados ao propósito comunicativo e ao público-alvo. Desta forma, buscou-se identificar parâmetros que possam estar associados a um maior sucesso das publicações (isto é, a um maior número de acessos) no website do IF Goiano.

Por conseguinte, empregou-se a biblioteca *Transformers*, da linguagem de programação Python, para acessar um modelo *BART* com suporte à língua portuguesa, com o propósito de realizar a sumarização dos textos correspondentes ao corpo das notícias. Na configuração do processo de sumarização, tornou-se necessário definir um tamanho médio para o texto resultante, isto é, o resumo gerado. Para tanto, utilizou-se como parâmetro o comprimento médio das descrições originais disponíveis no website. Os Códigos 28 e 29 apresentam em detalhe esse procedimento.

No Código 28, na linha 6, observa-se que o modelo selecionado para este trabalho foi o *bart-large-cnn*[12], desenvolvido pela Meta (antiga Facebook). Esse modelo foi ajustado a partir de um conjunto de dados provenientes do portal de notícias britânico CNN, sendo voltado, primordialmente, para o idioma inglês, embora ofereça suporte a outros idiomas, inclusive o português, utilizado neste estudo.

[12]Mais informações sobre este modelo podem ser encontradas em `https://huggingface.co/facebook/bart-large-cnn` . Acesso em 20/06/2024.

```
1    import pandas as pd
2    import numpy as np
3    from transformers import pipeline, BartForConditionalGeneration, BartTokenizer
4
5    # Define o modelo BART para sumarização
6    MODELO_BART = 'facebook/bart-large-cnn'
7    # Inicializa o pipeline de sumarização
8    sumarizador = pipeline(
9        task='summarization',
10       model=MODELO_BART
11   )
12   # Carrega o tokenizador específico do BART
13   tokenizador_bart = BartTokenizer.from_pretrained(MODELO_BART)
14   # Carrega o modelo BART para geração condicional
15   modelo_bart = BartForConditionalGeneration.from_pretrained(MODELO_BART)
```

Código 28: Configuração dos objetos para se utilizar o modelo BART *facebook/bart-large-cnn*.

Na linha 8, é construído o objeto *sumarizador*, responsável por receber o texto a ser processado e retornar o respectivo resumo gerado pelo modelo. Na linha 13, o objeto *tokenizador_bart* atua como um componente auxiliar encarregado de pré-processar os textos das notícias, convertendo-os para o formato interno compatível com o *BART*, o que se mostra especialmente útil no caso de textos de maior extensão. Por fim, na linha 15, o objeto *modelo_bart* disponibiliza o modelo principal do *BART* para a geração dos sumários, constituindo uma alternativa mais customizável ao objeto *sumarizador*.

No Código 29, deu-se continuidade à etapa de sumarização ao acessar, na linha 4, a coluna `corpo`, a qual contém o texto integral das notícias, presente no *data-frame* denominado `df`, previamente carregado a partir do banco de dados. Em seguida, aplicaram-se os objetos instanciados no código anterior, necessários para a representação interna do modelo, com o objetivo de gerar a nova descrição de cada notícia.

```
1  for indice in range(df.shape[0]):
2      # Tokeniza o texto do corpo da notícia para o modelo BART
3      tokens = tokenizador_bart(
4          df.loc[indice, 'corpo'],
5          max_length=1024,           # Limite máximo de tokens
6          return_tensors='pt',       # Retorna tensores do PyTorch
7          padding=True,              # Preenche textos curtos
8          truncation=True            # Trunca textos longos
9      )
10     # Gera o sumário usando o modelo BART
11     ids_sumario = modelo_bart.generate(
12         tokens['input_ids'],
13         num_beams=4,               # Número de feixes para busca
14         min_length=tamanho_minimo, # Tamanho mínimo do sumário
15         max_length=tamanho_maximo, # Tamanho máximo do sumário
16         early_stopping=True        # Para quando encontrar bom resultado
17     )
18     # Decodifica os tokens gerados em texto legível
19     texto_decodificado = tokenizador_bart.decode(
20         ids_sumario[0],
21         skip_special_tokens=True   # Remove tokens especiais
22     )
23     # Refina o sumário usando o pipeline de sumarização
24     texto_sumario = sumarizador(
25         texto_decodificado,
26         do_sample=False            # Geração determinística
27     )[0]['summary_text']
```

Código 29: Aplicação dos objetos de representação interna do modelo BART para se realizar a sumarização do corpo das notícias.

Nas linhas 4 a 8, o objeto `tokenizador_bart` foi configurado para receber como argumentos: (i) o texto a ser sumarizado; (ii) o limite máximo de *tokens* a ser retornado; (iii) o tipo de *tokens* de saída, neste caso, `pt`, correspondente aos tensores da biblioteca *PyTorch*; (iv) o preenchimento (*padding*); e (v) o truncamento (*truncation*), ativados, respectivamente, para complementar textos de menor extensão e truncar aqueles que excedem o limite estabelecido.

Para a configuração do modelo condicional *BART*, utilizaram-se, nas linhas 12 a 16, os seguintes argumentos: (i) os *tokens* gerados na etapa anterior; (ii) o número de feixes (*beams*) empregados na busca, isto é, as N melhores sequências consideradas em cada passo da geração; (iii) o comprimento mínimo do sumário; (iv) o comprimento

máximo do sumário; e (v) o encerramento antecipado (*early stopping*) ativado, de modo que o processo seja finalizado assim que um sumário satisfatório for encontrado.

De posse do sumário em formato codificado, empregou-se, nas linhas 19 a 22, o objeto `tokenizador_bart` para realizar a decodificação dos *tokens* gerados, por meio do método `decode`. Foram passados, como argumentos: (i) os *tokens* a serem decodificados; e (ii) o parâmetro `skip_special_tokens`, responsável por remover os *tokens* especiais que não estavam presentes na entrada original, mas que são utilizados internamente pelo modelo durante o processo de geração.

Por fim, nas linhas 24 a 27, aplica-se o objeto `sumarizador`, uma *pipeline* configurada especificamente para a tarefa de sumarização, sobre o texto decodificado, a fim de se obter o resultado final do processo. Ao invocar esse objeto, são passados como argumentos: (i) o texto decodificado; e (ii) o parâmetro `do_sample`, desativado para que o modelo realize uma busca determinística durante a geração, produzindo resultados mais previsíveis e coerentes para este tipo de aplicação.

A abordagem processa o texto em três etapas sequenciais: inicialmente, converte-se o texto em *tokens* numéricos, empregando truncamento inteligente para lidar com limites de tamanho; em seguida, gera-se os *tokens* do sumário por meio de busca em feixe (*beam search*), a fim de otimizar a qualidade da saída; e, por fim, realiza-se a decodificação desses *tokens* em texto legível, removendo *tokens* especiais.

Essa metodologia permite maior controle sobre os parâmetros de geração, possibilita o tratamento personalizado de textos extensos por meio do truncamento e assegura sumários mais coerentes graças à busca em feixe, resultando em saídas consistentes e adequadas para análises científicas comparativas.

7 RESULTADOS E DISCUSSÕES

Durante a implementação dos *scripts* Python para realizar as consultas ao banco de dados do projeto, avaliou-se quais informações seriam as mais adequadas para serem transformadas em gráficos, a fim de representar visualmente os resultados mais relevantes em relação aos objetivos do trabalho, e quais se destinariam apenas à exploração preliminar da base de dados.

7.1 QUANTIDADE DE ACESSOS

Inicialmente, elaborou-se um esquema de análise das características das publicações com o propósito de identificar o melhor dia da semana para a divulgação de notícias. A partir da observação dos dados coletados, verificou-se que **o dia com maior média de acessos**, durante o período monitorado, **foi quinta-feira**, conforme ilustrado na Figura 14.

Constatou-se, entretanto, uma discrepância expressiva entre quinta-feira e os demais dias da semana. A diferença na média de acessos variou de 16.913, quando comparada à segunda-feira (segundo maior valor médio), até 18.496, em relação ao domingo, que apresentou a menor média de acessos no período analisado.

Esse resultado sugere que, em uma ou mais quintas-feiras, determinadas notícias alcançaram um volume de acessos significativamente acima do padrão, distorcendo a média geral. Tal evidência indicou a necessidade de uma análise investigativa mais aprofundada, a qual foi realizada na sequência.

Após o cálculo da variação no número de acessos das notícias ao longo do período de coleta, constatou-se que um grupo específico de publicações não havia sido identificado pelo robô de software no dia anterior à ocorrência do valor *outlier* de acessos. Dessa forma, ao computar a diferença de acessos entre os dias 1º e 2 de fevereiro (total de acessos do dia 2 menos o total do dia 1), observou-se que essas notícias apresentavam valor nulo de acessos no dia 1, uma vez que ainda não haviam sido capturadas pelo sistema.

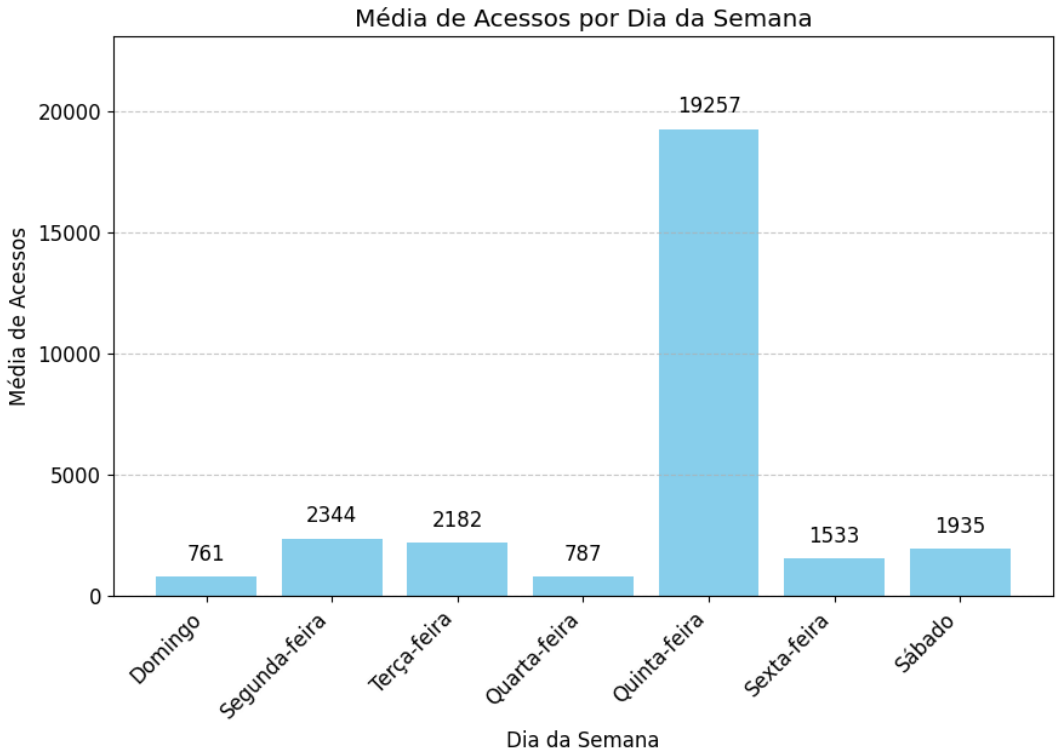

Figura 14: Média da quantidade de acessos por dia da semana.

Fonte: Elaborado pelo autor.

Assim, a variação de acessos foi interpretada pelo algoritmo como equivalente ao próprio total de acessos, o que resultou em valores significativamente discrepantes em relação aos demais registros. Embora ainda não seja possível determinar a razão pela qual essas notícias (aproximadamente 40 publicações) não tenham sido identificadas pelo robô de software antes da quinta-feira, 2 de fevereiro, esse fato contribuiu diretamente para a geração de valores *outlier* durante a modelagem dos dados.

Dessa forma, quinta-feira foi classificada como um *outlier*, em virtude da acentuada discrepância observada, apresentando um Z-score aproximado de 2,44. Tal valor revela que o número médio de acessos desse dia situa-se 2,44 desvios-padrão acima da média geral, caracterizando-se como uma anomalia estatisticamente relevante quando comparada aos demais dias. O Z-score, nesse contexto, expressa a quantidade de desvios-padrão que um valor se encontra distante da média.

Por conseguinte, ao desconsiderar as quintas-feiras e ordenar os dias da semana de forma decrescente pela quantidade média de acessos das notícias, obtém-se o gráfico apresentado na Figura 15. Observa-se que **segunda-feira registra a maior média de acessos**, atingindo mais de 2.300 visualizações diárias durante o período analisado.

Em seguida, aparecem terça-feira e sábado, com médias de 2.182 e 1.935 acessos, respectivamente, valores próximos entre si, com diferença de aproximadamente 200 acessos. Um aspecto interessante é que os dias segunda, terça, sexta-feira e sábado apresentaram médias de acessos superiores à quantidade de minutos existentes em um dia (24h × 60min = 1.440 minutos). Assim, pode-se inferir que, nesses dias, em média, **ao menos uma notícia foi visualizada a cada minuto** na plataforma do IF Goiano.

A seguir, com o objetivo de obter um novo *insight*, procedeu-se à análise das notícias de acordo com o assunto, agrupando-as segundo a classificação temática previamente estabelecida. Buscou-se, assim, identificar qual tema apresentou a maior média de acessos, ou seja, o assunto mais popular entre as publicações do site do IF Goiano.

Como pode ser observado no gráfico da Figura 16, o assunto **Editais** destacou-se como o mais popular, alcançando uma média de 1.485 acessos por notícia. Em seguida, aparecem **Processos Seletivos**, com 1.137 acessos médios, e **Ações de Extensão**, com 483. Observa-se que os dois primeiros temas alcançaram uma quantidade de

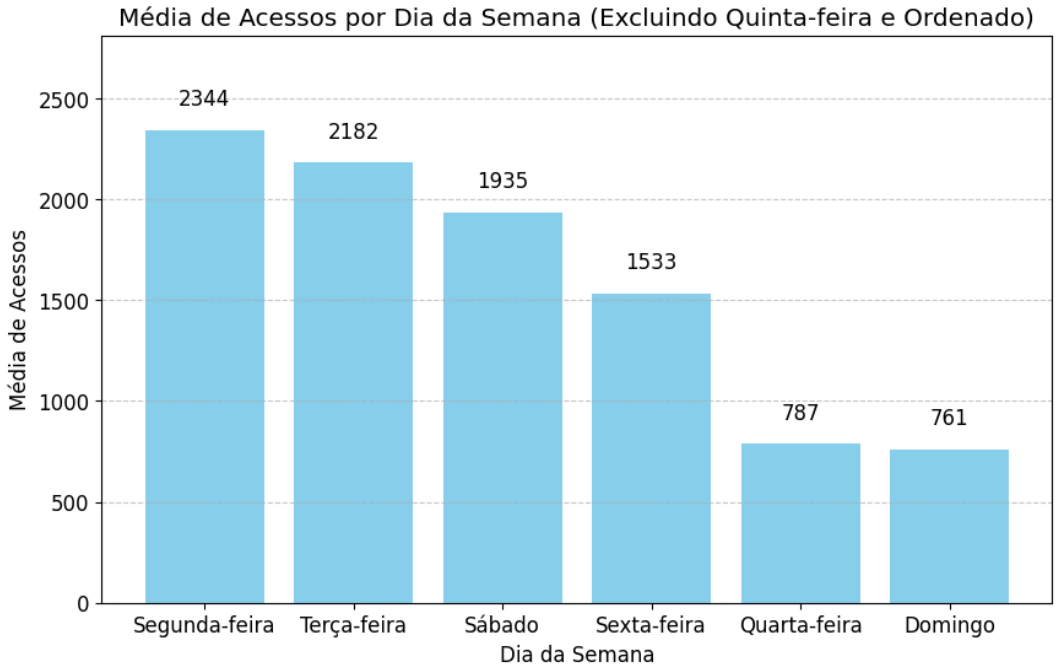

Figura 15: Média da quantidade de acessos por dia da semana, excluindo *outliers*.

Fonte: Elaborado pelo autor.

acessos bastante próxima, muito embora os editais, geralmente, abordam projetos de pesquisa, ensino e extensão, enquanto processos seletivos, abordam sobre oportunidades para alunos ingressarem na instituição.

Na análise seguinte, buscou-se avaliar a quantidade de acessos por assunto, com o objetivo de identificar qual dia da semana determinado assunto alcançou a maior quantidade total de acessos durante o período monitorado. A partir dessa correlação, torna-se possível inferir o dia mais propício para a publicação de notícias relacionadas a cada tema específico.

Para o tema Editais, o dia com a maior quantidade de acessos, e, portanto, o mais propício para a publicação de notícias dessa categoria, foi **quinta-feira**, com 19.435 acessos, seguido por segunda-feira, com 1.868, e por sexta-feira, com 1.520 acessos, conforme apresentado na Figura 17.

No entanto, como se pode observar, o valor obtido por quinta-feira é significativamente superior aos demais dias da semana, podendo ser considerado um *outlier*, ao atingir um escalar de aproximadamente 2,44 no cálculo do Z-score. A seguir, no gráfico apresentado na Figura 18, remove-se o *outlier* e identificam-se como dias mais propícios

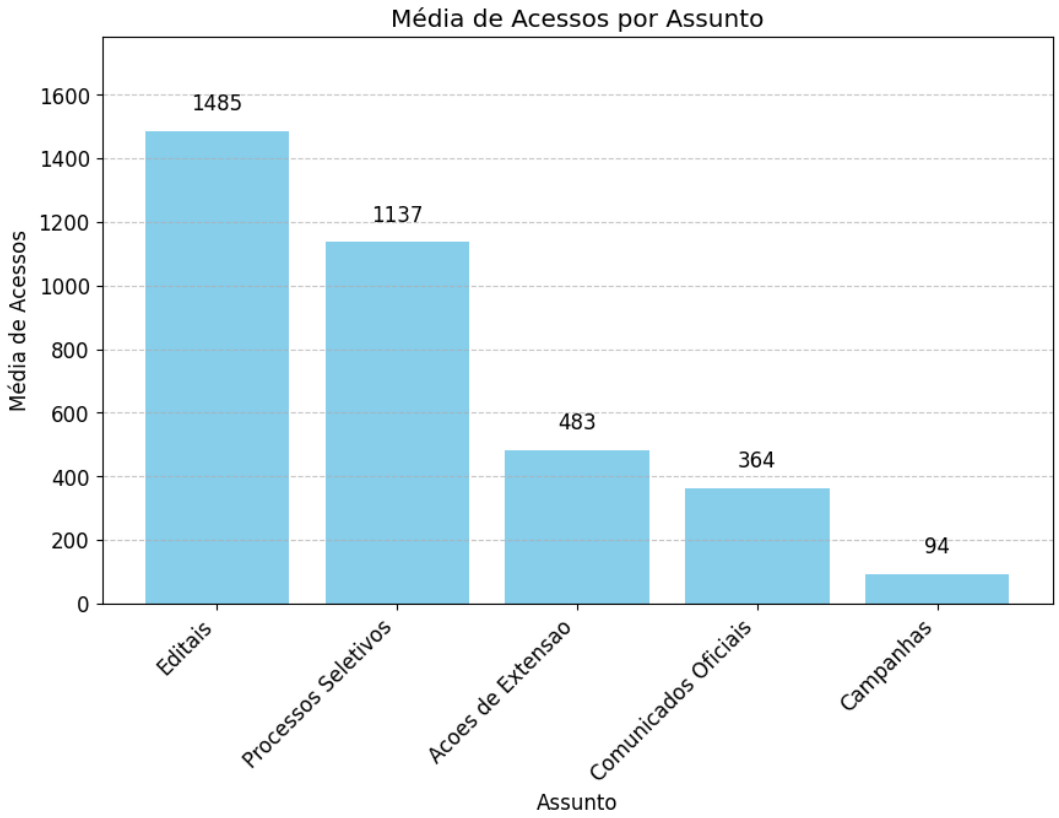

Figura 16: Média da quantidade de acessos pelo assunto da notícia.

Fonte: Elaborado pelo autor.

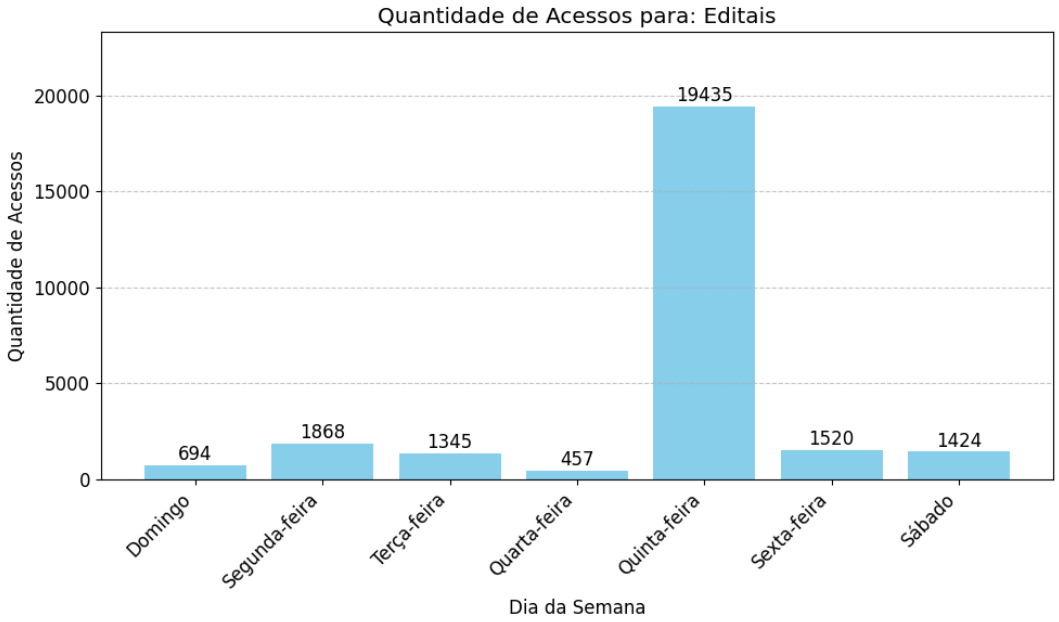

Figura 17: Quantidade de acessos por dia da semana para as notícias com assunto ”Editais”.

Fonte: Elaborado pelo autor.

para a publicação de notícias desse tema: **segunda-feira**, com 1.868 acessos, sexta-feira, com 1.520, e sábado, com 1.424 acessos.

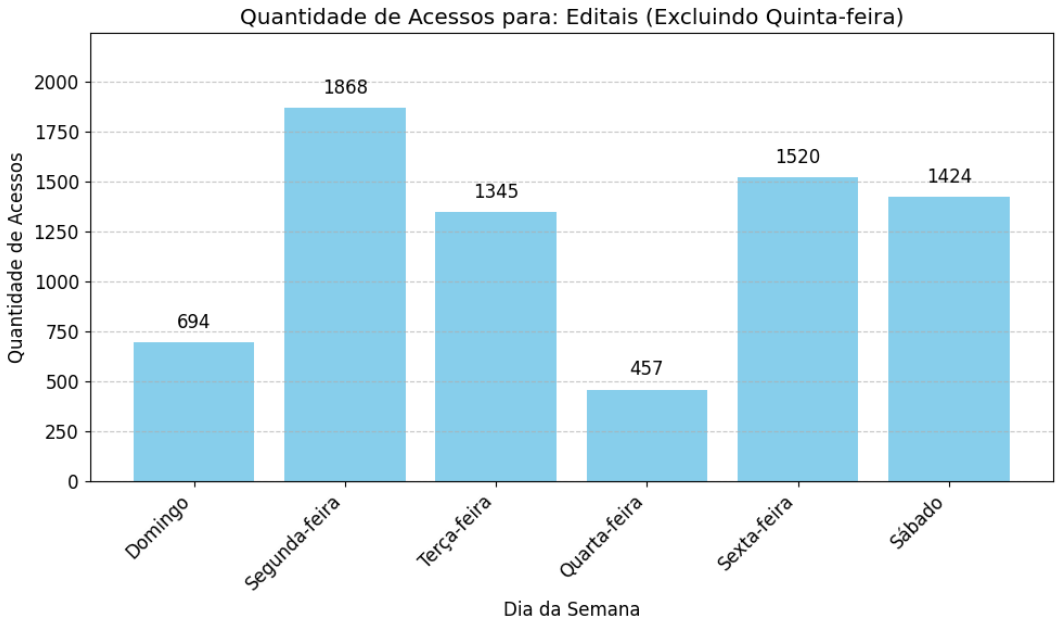

Figura 18: Quantidade de acessos por dia da semana para as notícias com assunto ”Editais”, removendo *outliers*.

Fonte: Elaborado pelo autor.

Para o tema Processos Seletivos, o dia com a maior quantidade de acessos, e, portanto, o mais propício para a publicação de notícias dessa categoria, foi **quinta-feira**, com 11.858 acessos, seguido por segunda-feira, com 3.017, e por sexta-feira, com 2.055 acessos, conforme apresentado na Figura 19.

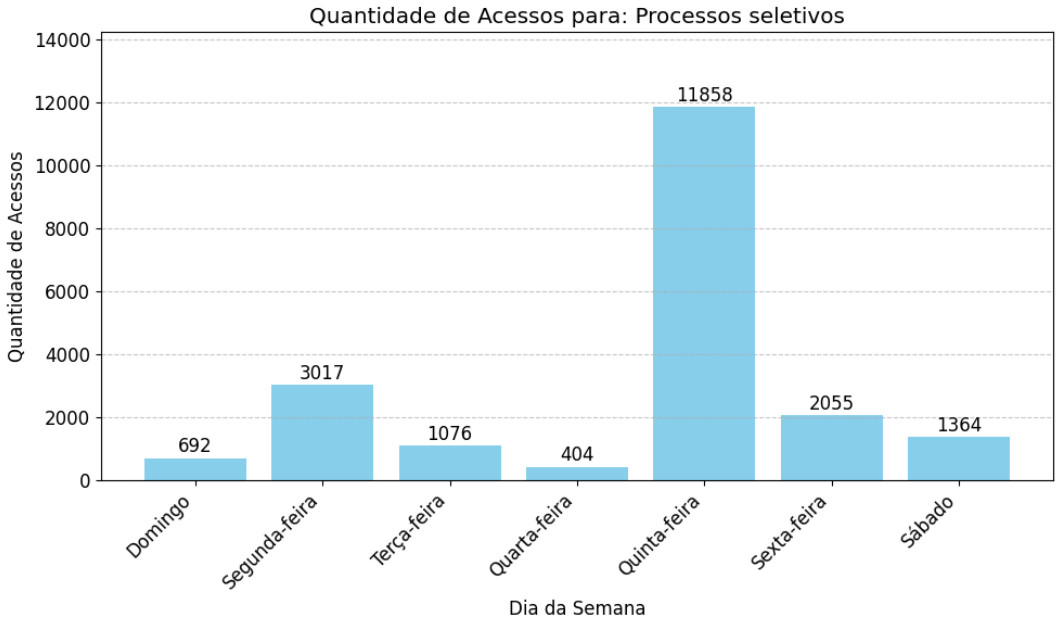

Figura 19: Quantidade de acessos por dia da semana para as notícias com assunto ”Processos Seletivos”.

Fonte: Elaborado pelo autor.

No entanto, como se pode observar, o valor obtido por quinta-feira é significativamente superior aos demais dias da semana, podendo ser considerado um *outlier*, ao atingir um escalar de aproximadamente 2,39 no cálculo do Z-score. A seguir, no gráfico apresentado na Figura 20, remove-se o *outlier* e identificam-se como dias mais propícios para a publicação de notícias desse tema: **segunda-feira**, com 3.017 acessos, sexta-feira, com 2.055, e sábado, com 1.364 acessos.

Para o tema Ações de Extensão, o dia com a maior quantidade de acessos, e, portanto, o mais propício para a publicação de notícias dessa categoria, foi **sábado**, com 2.324 acessos, seguido por quinta-feira, com 2.157, e segunda-feira, com 1.299 acessos, conforme apresentado na Figura 21.

Para o tema Comunicados Oficiais, o dia com a maior quantidade de acessos, e, portanto, o mais propício para a publicação de notícias dessa categoria, foi **quinta-feira**, com 4.823 acessos, seguido por segunda-feira, com 463, e por sábado, com 378 acessos, conforme apresentado na Figura 22.

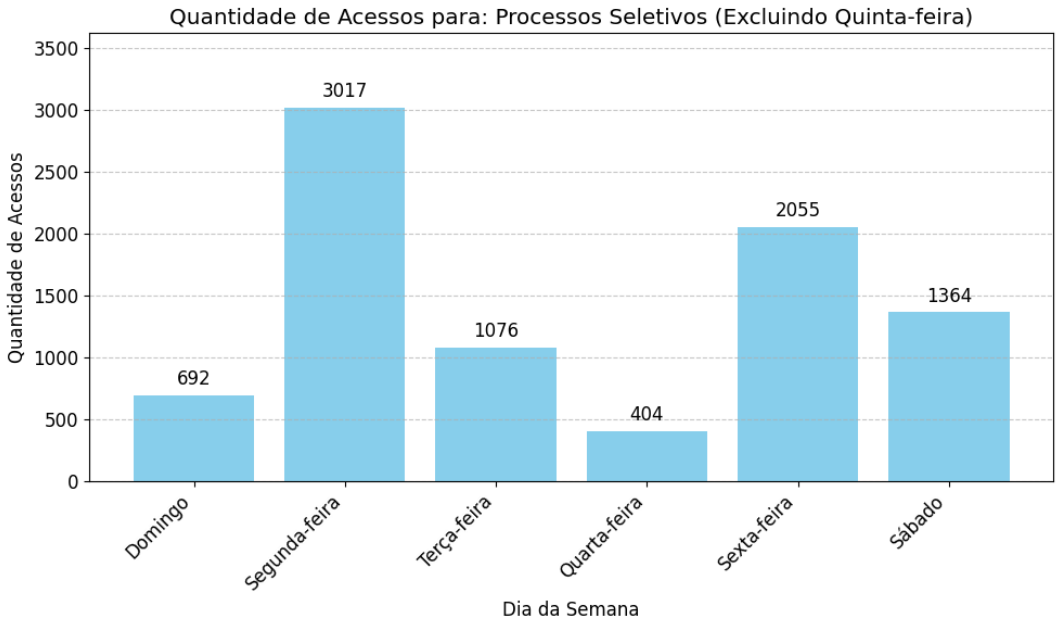

Figura 20: Quantidade de acessos por dia da semana para as notícias com assunto ”Processos Seletivos”, removendo *outliers*.

Fonte: Elaborado pelo autor.

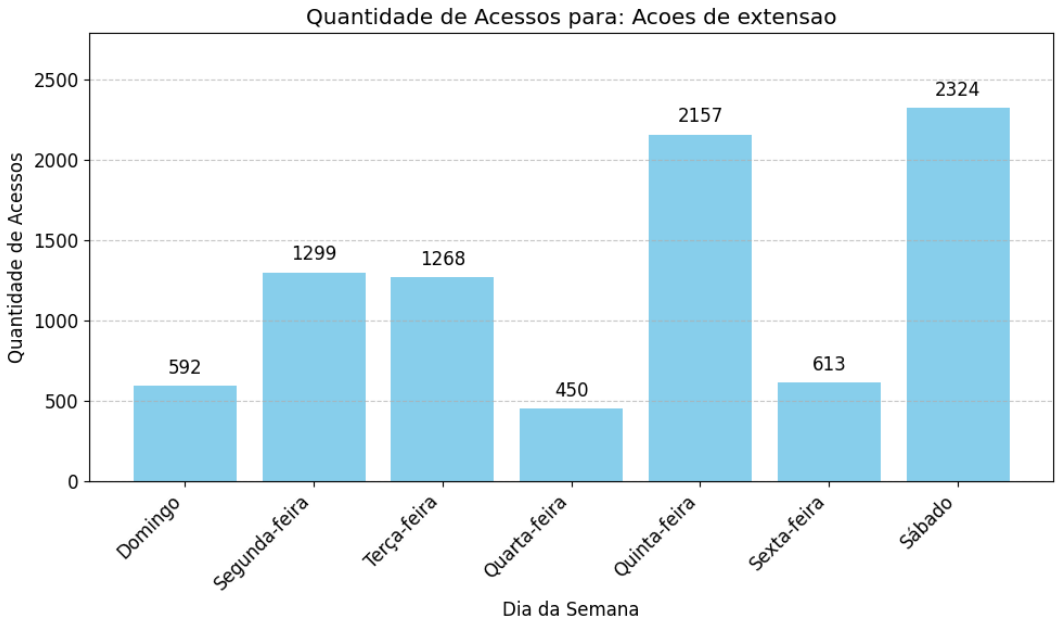

Figura 21: Quantidade de acessos por dia da semana para as notícias com assunto ”Ações de Extensão”.

Fonte: Elaborado pelo autor.

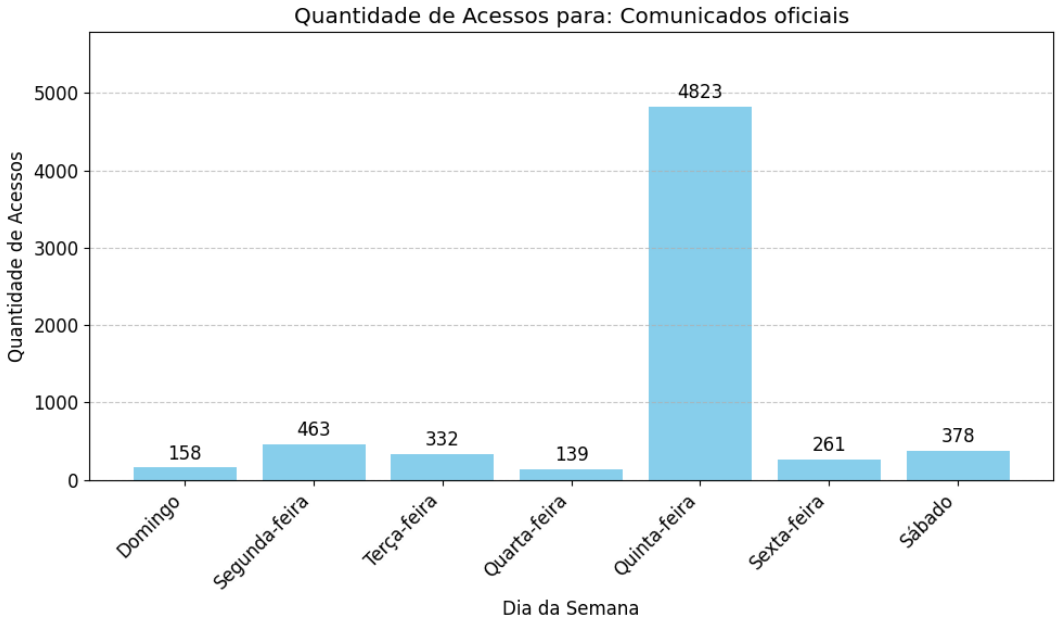

Figura 22: Quantidade de acessos por dia da semana para as notícias com assunto ”Comunicados Oficiais”.

Fonte: Elaborado pelo autor.

No entanto, como se pode observar, o valor obtido por quinta-feira é significativamente superior aos demais dias da semana, podendo ser considerado um *outlier*, ao atingir um escalar de aproximadamente 2,44 no cálculo do Z-score. A seguir, no gráfico apresentado na Figura 23, remove-se o *outlier* e identificam-se como dias mais propícios para a publicação de notícias desse tema: **segunda-feira**, com 463 acessos, sábado, com 378, e terça-feira, com 332 acessos.

Para o tema Campanhas, o dia com a maior quantidade de acessos, e, portanto, o mais propício para a publicação de notícias dessa categoria, foi **segunda-feira**, com 385 acessos, seguido por terça-feira, com 344, e sábado, com 317 acessos, conforme apresentado na Figura 24.

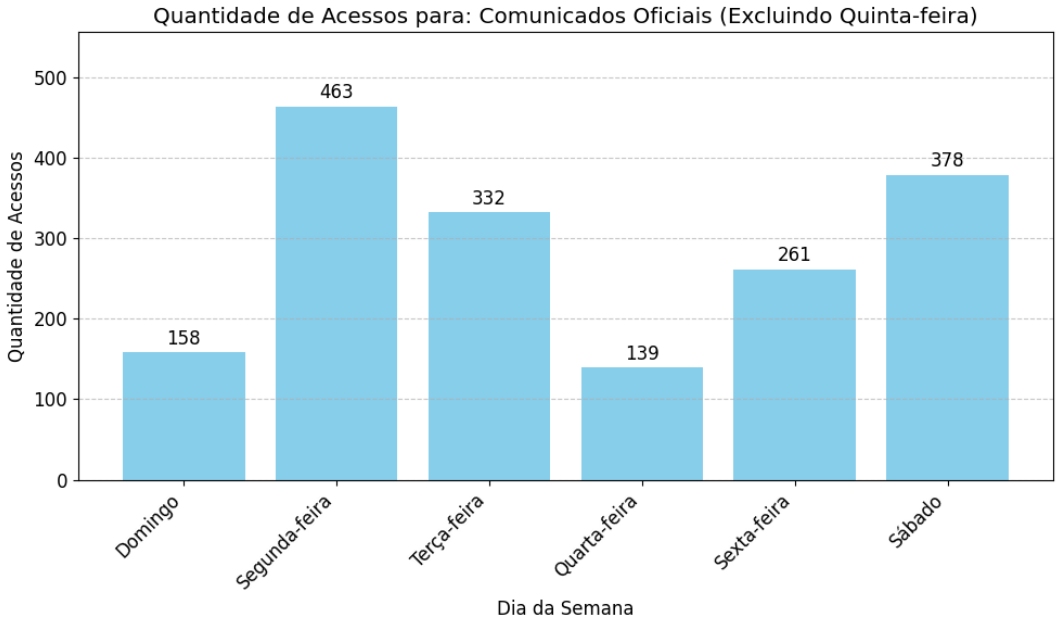

Figura 23: Quantidade de acessos por dia da semana para as notícias com assunto ”Comunicados Oficiais”, removendo *outliers*.

Fonte: Elaborado pelo autor.

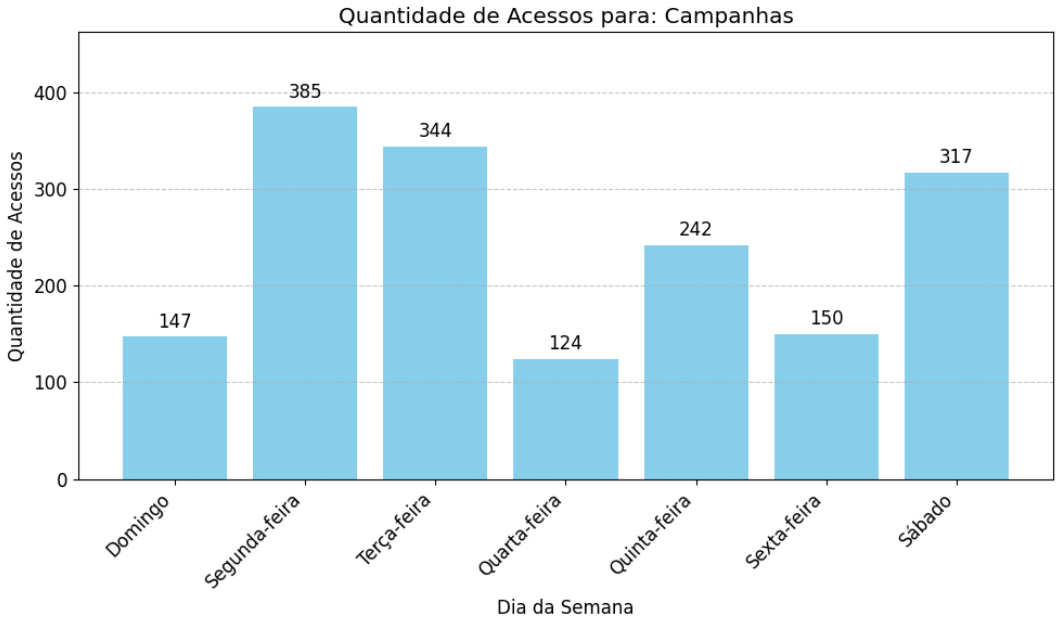

Figura 24: Quantidade de acessos por dia da semana para as notícias com assunto ”Campanhas”.

Fonte: Elaborado pelo autor.

Assim, os dias da semana mais propícios para a publicação de notícias, de acordo com o tema específico, encontram-se listados na Tabela 13. **Segunda-feira** desponta como o dia da semana mais comum na tabela, apresentando uma soma total de

acessos das notícias mais elevada, ou pelo menos entre os três maiores valores, no caso das notícias com tema Ações de Extensão, entre todos os dias da semana. Outro dia a ser destacado é **sábado**, que também aparece em todas as classificações de top 3 dias com mais acessos em todos os temas.

Tabela 13: Os dias da semana mais propícios para se publicar uma notícia no site do IF Goiano, de acordo com o assunto.

Assunto	Dia com melhor desempenho	Dia com maior número de acessos
Editais	Segunda-feira, sexta-feira e sábado	Quinta-feira
Processos Seletivos	Segunda-feira, sexta-feira e sábado	Quinta-feira
Ações de Extensão	Sábado, quinta-feira e segunda-feira	Sem outlier
Comunicados Oficiais	Segunda-feira, sábado e terça-feira	Quinta-feira
Campanhas	Segunda-feira, terça-feira e sábado	Sem outlier

Fonte: Elaborado pelo autor.

Como visto anteriormente, quinta-feira apresentou a maior variação de acessos dentre todos os dias da semana no período analisado, correspondente à etapa de coleta e monitoramento das notícias no site do IF Goiano. Nesse dia, o volume total ultrapassou 38.000 acessos, com média superior a 19.000 visualizações.

De forma mais específica, na quinta-feira, 02 de fevereiro de 2023, observou-se um pico expressivo de acessos nas notícias relacionadas aos assuntos ações de extensão, comunicados oficiais e editais. Esse comportamento destacou uma marca significativamente superior à registrada nos demais dias, indicando uma concentração atípica de interesse do público nessas publicações.

Após realizar uma investigação mais aprofundada, observou-se que esse aumento abrupto de acessos foi provocado por um conjunto de 40 notícias que não haviam sido identificadas pelo robô de software responsável pela coleta automatizada no dia anterior, isto é, na quarta-feira, 01 de fevereiro de 2023. Em razão dessa falha de detecção, os acessos totais dessas notícias foram contabilizados integralmente na quinta-feira, já que, no dia anterior, seus valores constavam como zero (total – 0 = total).

Ainda não foi possível esclarecer a causa da não localização dessas notícias pelo sistema de coleta, sobretudo porque suas datas de publicação são anteriores ao episódio. Em sua maioria, elas correspondem aos meses de dezembro e janeiro dos anos de 2024 e 2025, respectivamente, o que torna o ocorrido ainda mais incomum e carecendo de investigação complementar.

Conjectura-se que a equipe de manutenção do site do IF Goiano realize “ativações” e “desativações” de determinadas notícias no repositório institucional, considerando fatores técnicos como espaço de armazenamento, tráfego no servidor e rotinas de backup, bem como possíveis políticas internas de publicação e organização de conteúdo. Essa hipótese pode explicar a indisponibilidade temporária das notícias no momento da coleta automatizada.

7.2 NUVEM DE PALAVRAS POR ASSUNTO

Seguindo nas análises e aproveitando o conteúdo textual das notícias, especificamente os campos título e descrição, elaborou-se algumas nuvens de palavras segmentadas por assunto: editais, processos seletivos, ações de extensão, comunicados oficiais e campanhas. Esse tipo de visualização permite identificar, de forma intuitiva, quais termos são mais recorrentes dentro de um conjunto textual específico, evidenciando padrões linguísticos e elementos característicos de cada categoria. A seguir, na Figura 25, apresenta-se a nuvem de palavras referente ao assunto editais.

Figura 25: Nuvem de palavras das notícias com assunto *editais*.

Fonte: Elaborado pelo autor.

A nuvem de palavras revela que o assunto editais está diretamente associado ao ciclo acadêmico do IF Goiano, abrangendo as etapas fundamentais de divulgação, inscrição e, sobretudo, a publicação do resultado final dos processos. Observa-se que os editais regulam uma ampla variedade de oportunidades destinadas a estudantes, como bolsas, mestrado e doutorado, e também a servidores.

Além disso, a recorrência de nomes de meses sugere a existência de um calendário contínuo de seleções, atualizações e prorrogações de prazos ao longo de todo o ano, o que evidencia a dinâmica permanente dessas atividades institucionais.

A nuvem de palavras na Figura 26, revela que o assunto processos seletivos está diretamente associado ao ciclo de admissão e ofertas do IF Goiano, sendo marcado pelas etapas de inscrições, seleção e, sobretudo, pela publicação do resultado final. Os processos seletivos regulamentam vagas e bolsas para uma ampla gama de cursos e níveis, incluindo mestrado e doutorado, além de envolverem a participação de servidores.

Figura 26: Nuvem de palavras das notícias com assunto *processos seletivos*.

Fonte: Elaborado pelo autor.

A elevada frequência de menções a meses e a termos temporais como dia e prazo sugere a existência de um calendário institucional intenso e contínuo, que mantém essas atividades dinâmicas ao longo de todo o ano.

A nuvem de palavras sobre ações de extensão, apresentada na Figura 27, revela que essa dimensão institucional do IF Goiano é marcada predominantemente pela realização de eventos e programas que materializam o conceito de Extensão. O foco recai sobre a execução de projetos e encontros voltados a um público diverso, que abrange estudantes, servidores e a instituição como um todo.

Figura 27: Nuvem de palavras das notícias com assunto *ações de extensão*.

Fonte: Elaborado pelo autor.

Termos como formação e inovação indicam que as ações têm como objetivo o desenvolvimento e a transferência de conhecimento. A menção recorrente a diversos campi, como Rio Verde, Ceres e Urutaí, bem como o uso frequente do termo dia, confirma a intensa e descentralizada atividade extensionista, que se mostra essencial para a interação do IF Goiano com a comunidade.

A nuvem de palavras apresentada na Figura 28, revela que o assunto ”comunicados oficiais”está diretamente associado à estrutura de governança e à transparência institucional do IF Goiano. O fluxo de informação é centralizado pela Instituição, com destaque para a Reitoria e o Conselho, e é direcionado principalmente aos servidores e à comunidade.

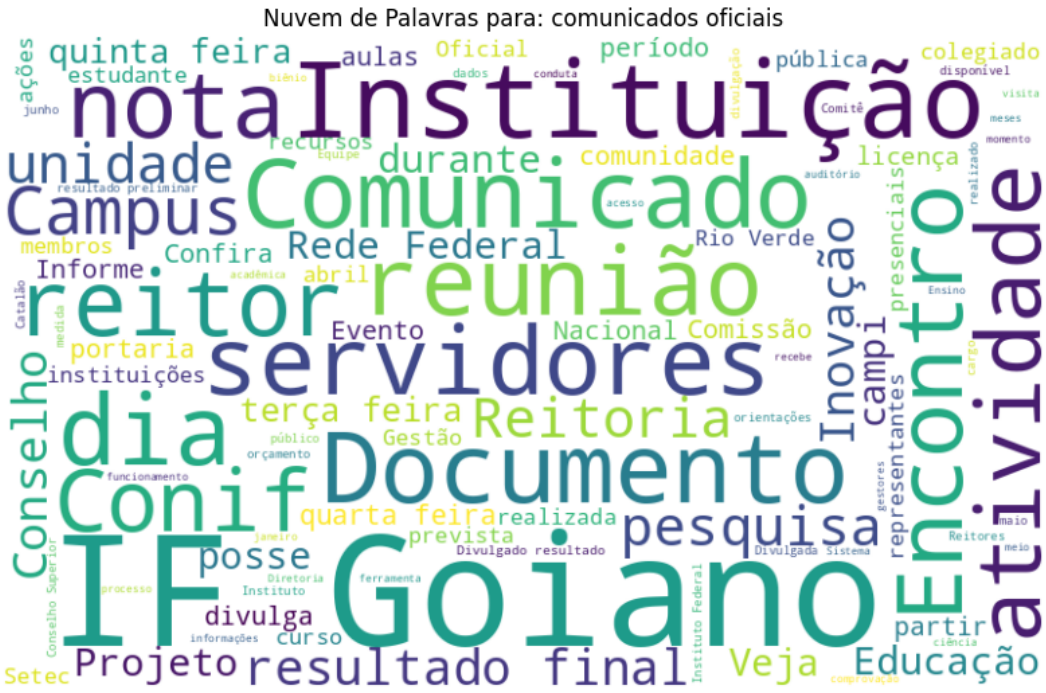

Figura 28: Nuvem de palavras das notícias com assunto *comunicados oficiais*.

Fonte: Elaborado pelo autor.

Os comunicados e notas oficiais assumem a forma de Documento ou Portaria, regulando assuntos essenciais como a realização de reuniões, a gestão de projetos nas áreas de Educação, Pesquisa e Inovação e a divulgação de resultados finais de processos. A expressiva presença de dias da semana e a recorrência de meses indicam um calendário administrativo intenso e contínuo, evidenciando a dinâmica permanente da gestão institucional.

A nuvem de palavras apresentada na Figura 29 revela que o assunto campanhas no IF Goiano está intimamente ligado à promoção de eventos, ações e atividades voltadas à Educação e à Inovação. O núcleo da comunicação se concentra na própria noção de Campanha, realizada pela Instituição em articulação com a Rede Federal e outras unidades.

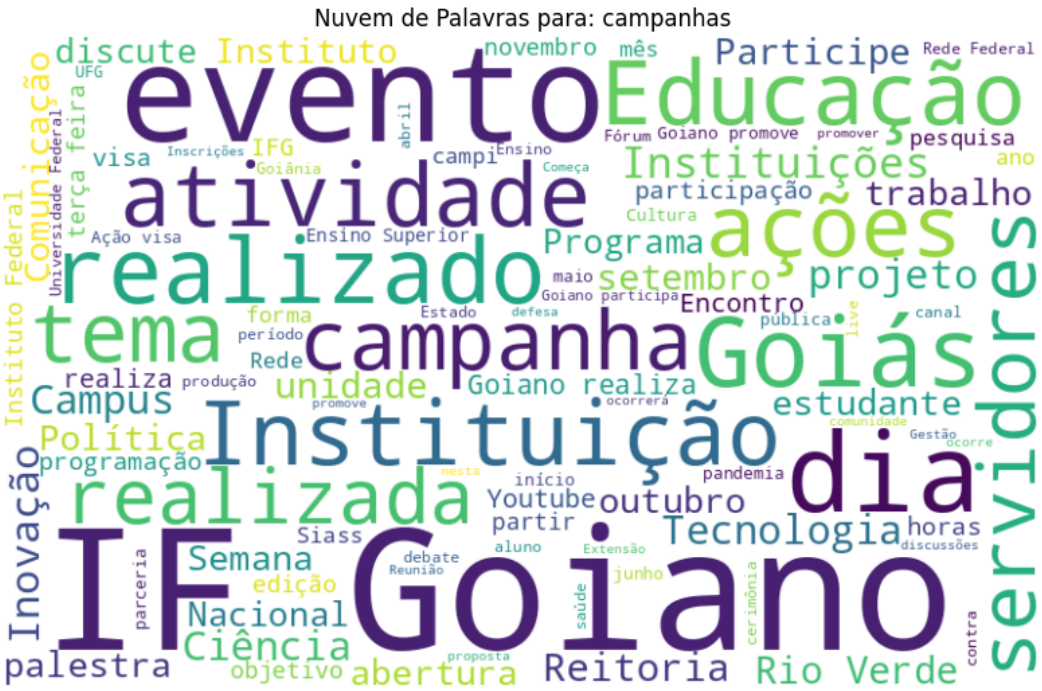

Figura 29: Nuvem de palavras das notícias com assunto *campanhas*.

Fonte: Elaborado pelo autor.

O público-alvo predominante abrange estudantes e servidores, com o propósito de estimular participação e mobilização. A recorrência de meses, dias e referências à Semana Nacional de Ciência e Tecnologia evidencia a presença de um calendário contínuo de iniciativas, além de destacar a importância da divulgação por meio de canais institucionais como o YouTube.

7.3 QUANTIDADE DE IMAGENS E SUA INFLUÊNCIA NA POPULARIDADE

Na análise seguinte, procurou-se investigar o conteúdo das notícias considerando a quantidade de imagens e sua influência na popularidade, levando em conta o dia da semana como fator de maior impacto. Dessa forma, foi possível identificar até quantas imagens no corpo da notícia tendem a contribuir para uma maior quantidade de acessos no site do IF Goiano. Parte-se do pressuposto de que uma maior riqueza de recursos

visuais, como imagens, pode atrair mais a atenção do leitor, incentivando-o a clicar na notícia e a consumi-la integralmente.

A quantidade de imagens presentes em uma notícia no site do IF Goiano variou entre 1 (uma) e 8 (oito) imagens. Iniciando pela categoria de notícias com 1 (uma) imagem, observa-se, conforme apresentado na Figura 30, que estas apresentam uma média de acessos bastante semelhante ao agrupamento geral de notícias por dia da semana. Tal comportamento decorre do fato de que a maioria das publicações se enquadra nessa categoria, isto é, contém apenas uma imagem em seu corpo.

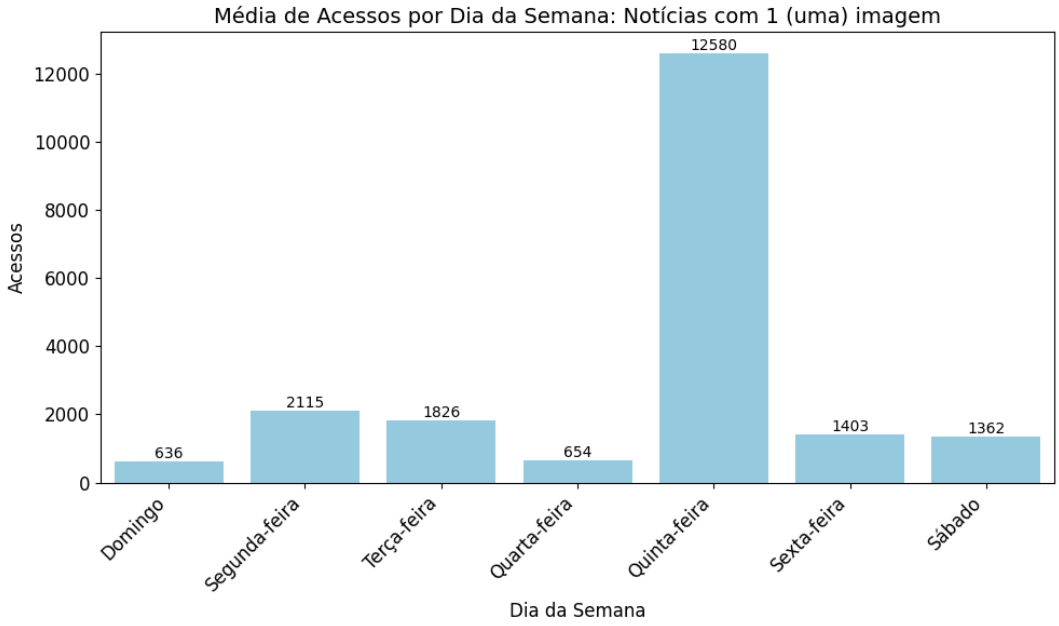

Figura 30: Média de acessos por dia da semana para as notícias com 1 (uma) imagem.

Fonte: Elaborado pelo autor.

Como observado nos gráficos anteriores, quinta-feira apresentou a maior quantidade média de acessos, podendo, contudo, ser considerada um *outlier*. Assim, ao desconsiderar esse dia da semana, obtém-se o gráfico apresentado na Figura 31. Constata-se, portanto, que **segunda-feira** é o dia com maior média de acessos para as notícias com 1 (uma) imagem, totalizando 6.344 acessos médios, seguida por sexta-feira, com 4.209, e por sábado, com 4.086 acessos médios.

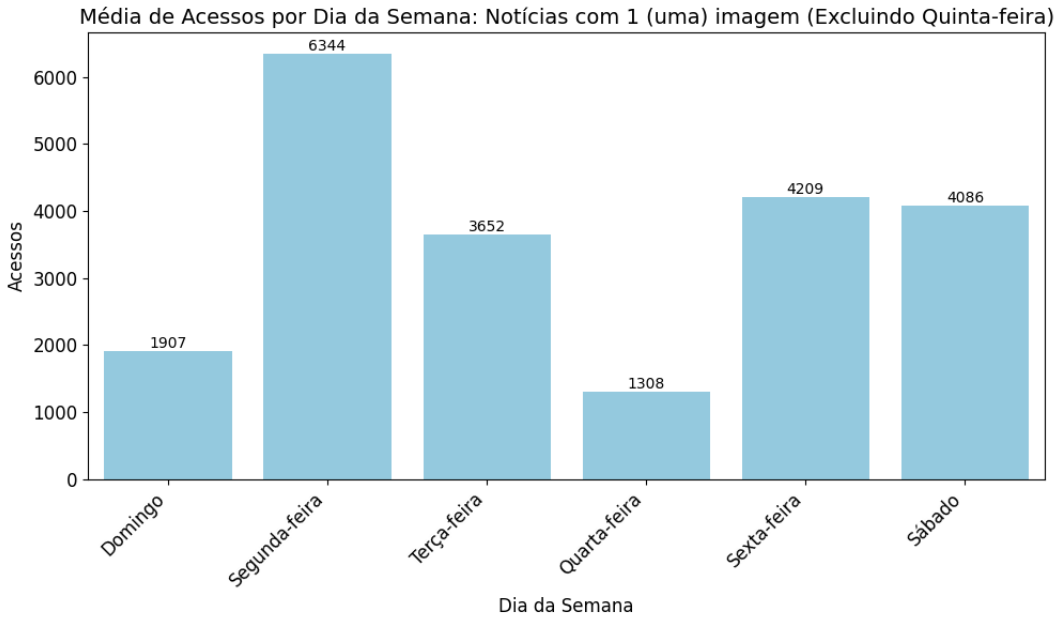

Figura 31: Média de acessos por dia da semana para as notícias com 1 (uma) imagem, excluindo *outliers*.

Fonte: Elaborado pelo autor.

Seguindo para o próximo agrupamento, correspondente às notícias com 2 (duas) imagens, observa-se, conforme ilustrado no gráfico da Figura 32, que **sábado** desponta como o dia da semana com a maior média de acessos, atingindo 522 acessos médios, seguido por terça-feira, com 294, e por quinta-feira, com 220 acessos médios.

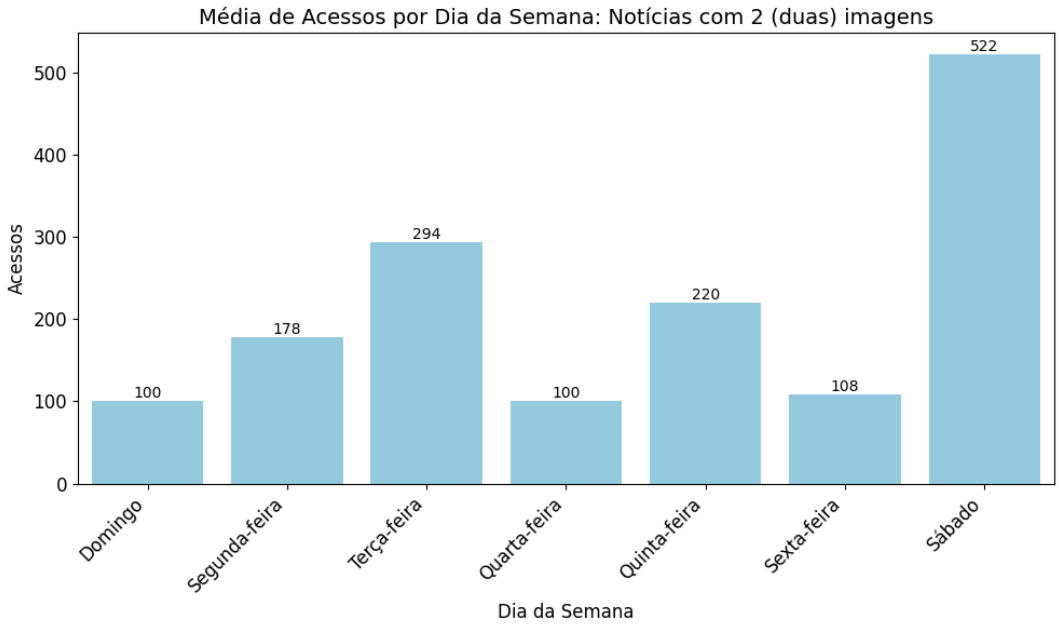

Figura 32: Média de acessos por dia da semana para as notícias com 2 (duas) imagens.

Fonte: Elaborado pelo autor.

Para as notícias com 3 (três) imagens, conforme ilustrado no gráfico da Figura 33, o dia da semana com a maior média de acessos foi **terça-feira**, com 32 acessos médios, seguido por segunda-feira, com 29, e por sábado, com 27 acessos médios.

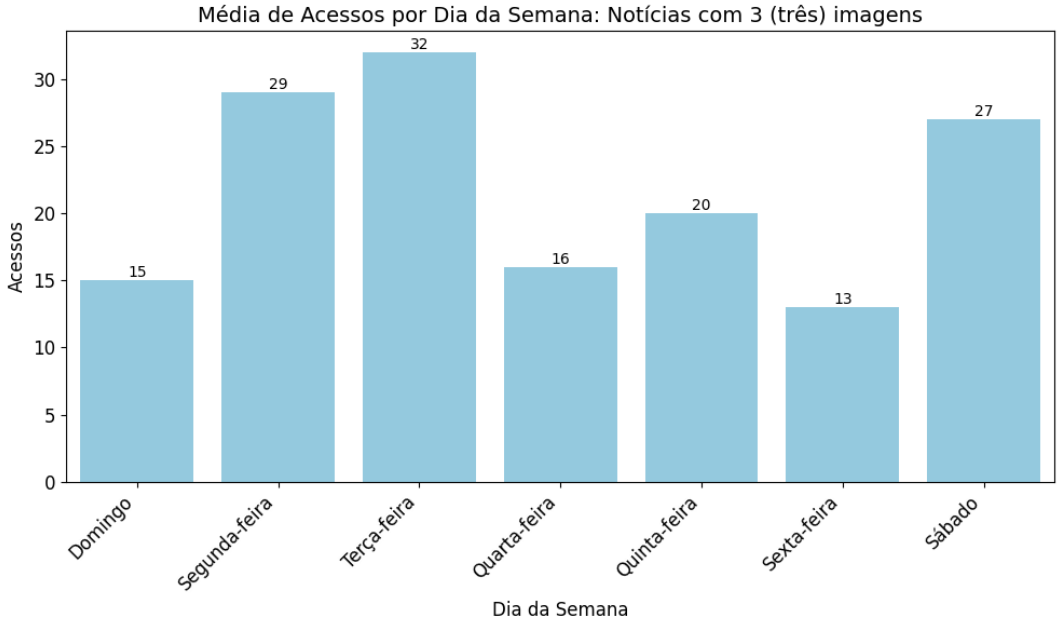

Figura 33: Média de acessos por dia da semana para as notícias com 3 (três) imagens.

Fonte: Elaborado pelo autor.

Para as notícias com 4 (quatro) imagens, conforme ilustrado no gráfico da Figura 34, o dia da semana com a maior média de acessos foi **terça-feira**, com 11 acessos médios, seguido por segunda-feira e sábado, ambos com 10 acessos médios.

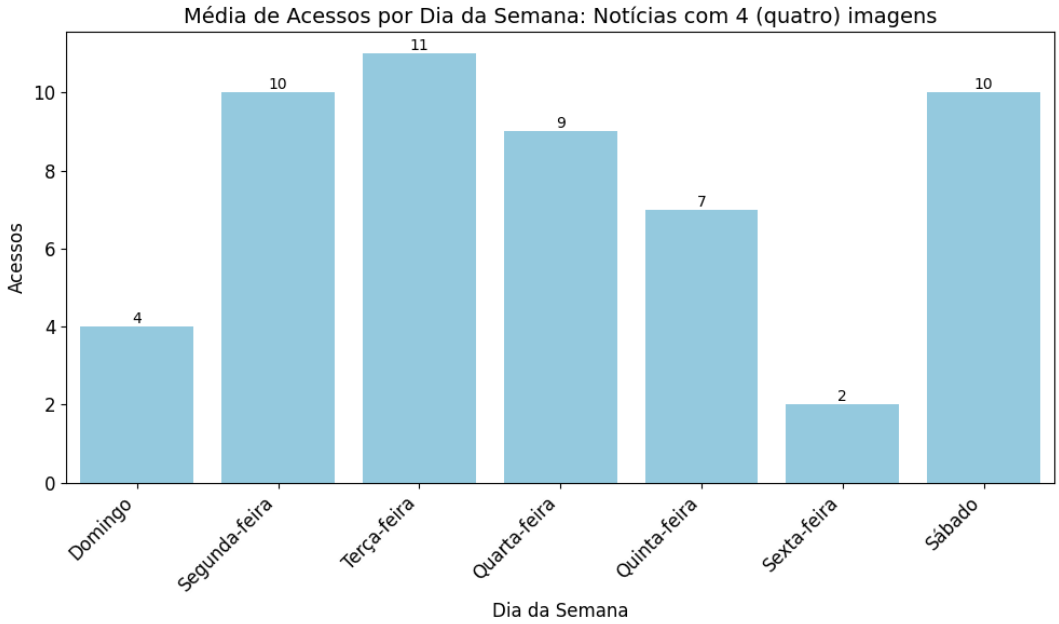

Figura 34: Média de acessos por dia da semana para as notícias com 4 (quatro) imagens.

Fonte: Elaborado pelo autor.

Para as notícias com 5 (cinco) imagens, conforme ilustrado no gráfico da Figura 35, os dias da semana com a maior média de acessos foram **terça-feira** e **sábado**, ambos com 7 acessos médios, seguidos por segunda-feira e quinta-feira, com 4 acessos médios.

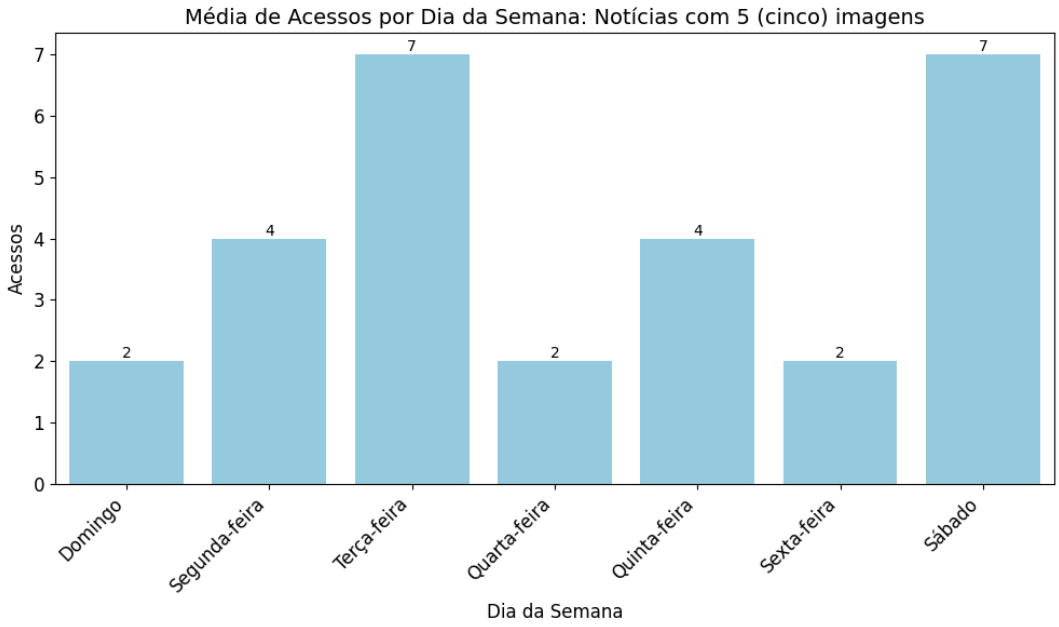

Figura 35: Média de acessos por dia da semana para as notícias com 5 (cinco) imagens.

Fonte: Elaborado pelo autor.

Para as notícias com 6 (seis) imagens, conforme ilustrado no gráfico da Figura 36, o dia da semana com a maior média de acessos foi **terça-feira**, com 7 acessos médios, seguido por segunda-feira, quarta-feira e quinta-feira, todas com 6 acessos médios.

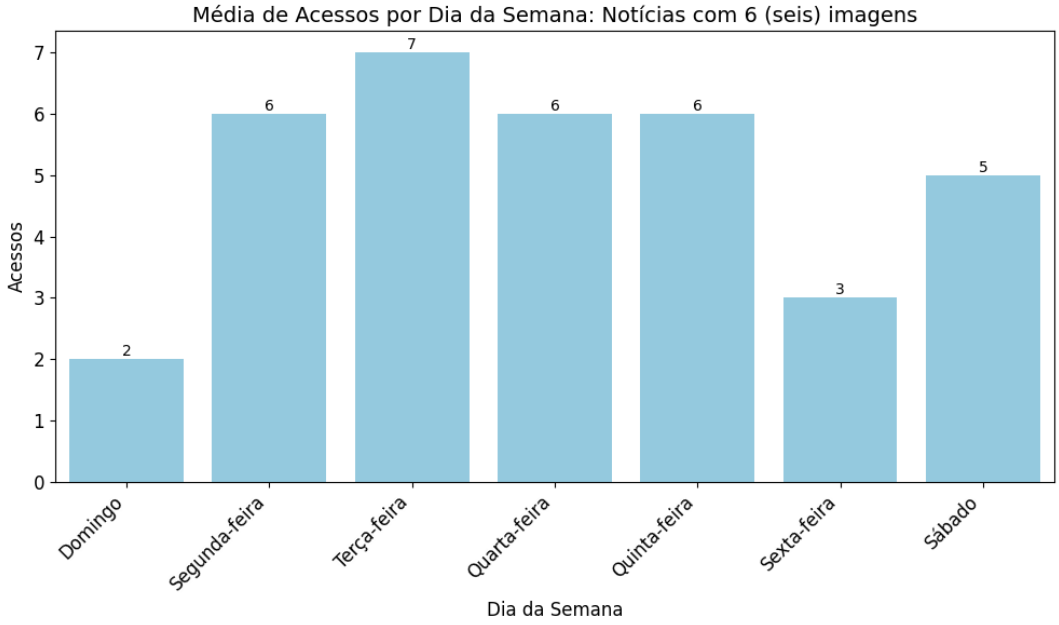

Figura 36: Média de acessos por dia da semana para as notícias com 6 (seis) imagens.

Fonte: Elaborado pelo autor.

Para as notícias com 7 (sete) imagens, conforme ilustrado no gráfico da Figura 37, o dia da semana com a maior média de acessos foi **terça-feira**, com 2 acessos médios, seguido por segunda-feira e quinta-feira, ambas com 1 acesso médio.

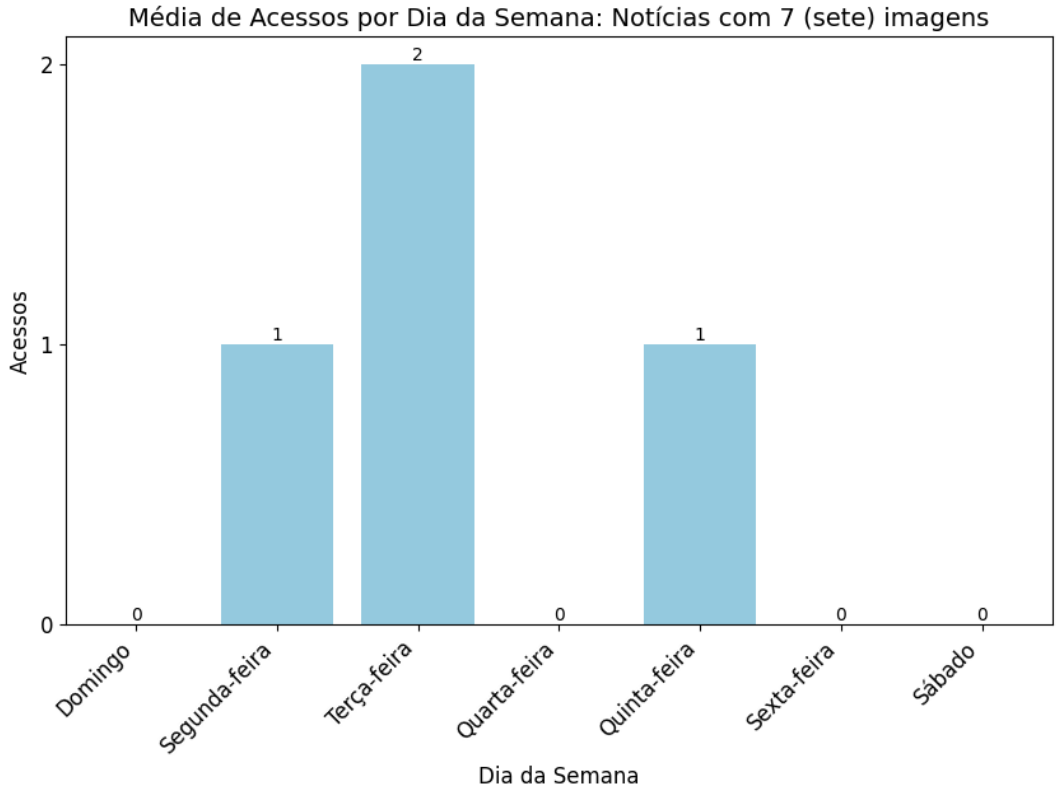

Figura 37: Média de acessos por dia da semana para as notícias com 7 (sete) imagens.

Fonte: Elaborado pelo autor.

Para as notícias com 8 (oito) imagens, conforme ilustrado no gráfico da Figura 38, o dia da semana com a maior média de acessos foi **terça-feira**, com 3 acessos médios, seguida por domingo, segunda-feira e sábado, todos com 2 acessos médios.

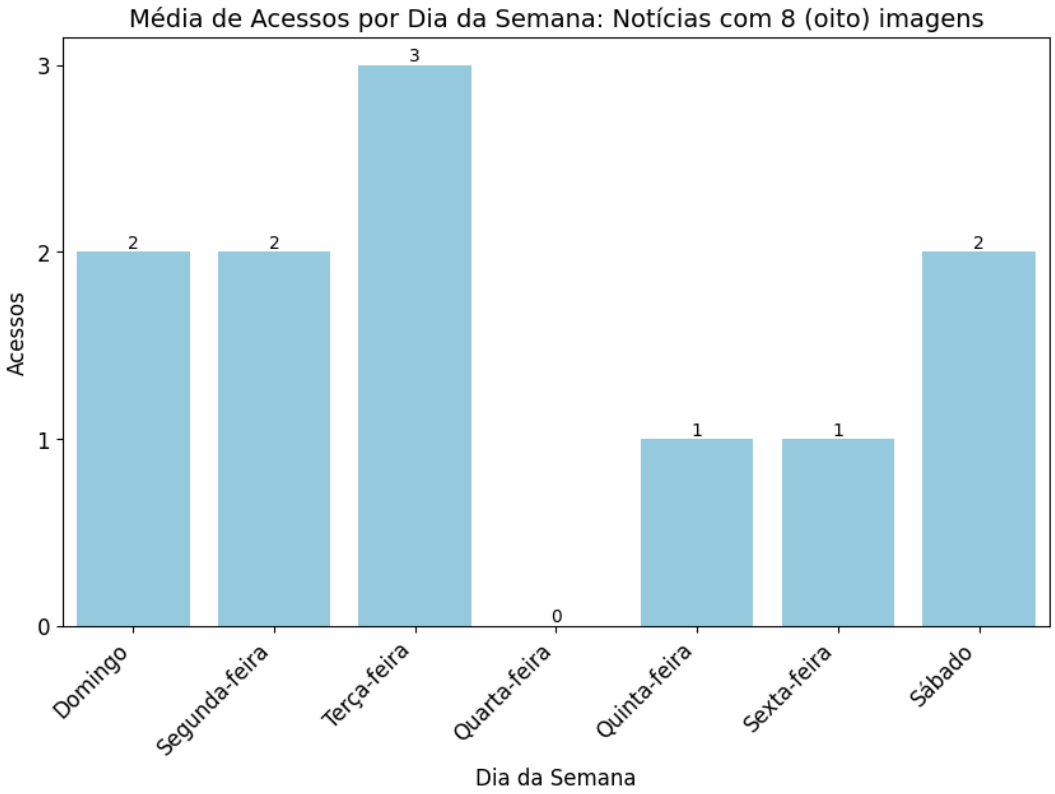

Figura 38: Média de acessos por dia da semana para as notícias com 8 (oito) imagens.

Fonte: Elaborado pelo autor.

Assim, os dias da semana mais propícios para a publicação de notícias, de acordo com a quantidade de imagens presentes em seu corpo, encontram-se listados na Tabela 14. **Terça-feira** destaca-se como o dia mais favorável para se publicar uma notícia no site do IF Goiano, apresentando o maior número de ocorrências entre as três primeiras posições de acessos médios nas diferentes quantidades de imagens. Outro dia a ser ressaltado é **segunda-feira**, que aparece em quase todas as classificações entre os três dias com maior média de acessos, indicando consistência na audiência independentemente do número de imagens.

As notícias também podem ser estratificadas conforme a presença ou ausência de uma **imagem de perfil**, isto é, uma imagem exibida na capa ou cabeçalho da notícia, visualizada primeiramente pelo leitor ao acessar o site do IF Goiano. Essa imagem funciona como uma porta de entrada para o conteúdo a ser explorado, devendo, portanto, apresentar um visual coerente com o tema abordado e suficientemente atrativo para captar a atenção do público. Assim, há notícias **com imagem de perfil** e notícias **sem imagem de**

Tabela 14: Dias da semana mais propícios para se publicar uma notícia no site do IF Goiano, de acordo com a quantidade de imagens.

Quantidade de Imagens	Dia com melhor desempenho	Dia com maior número de acessos
1 imagem	Segunda-feira, sexta-feira e sábado	Quinta-feira
2 imagens	Sábado, terça-feira e quinta-feira	Sem outlier
3 imagens	Terça-feira, segunda-feira e sábado	Sem outlier
4 imagens	Terça-feira, segunda-feira e sábado	Sem outlier
5 imagens	Terça-feira, sábado, segunda-feira e quinta-feira	Sem outlier
6 imagens	Terça-feira, segunda-feira, quarta-feira e quinta-feira	Sem outlier
7 imagens	Terça-feira, segunda-feira e quinta-feira	Sem outlier
8 imagens	Terça-feira, domingo, segunda-feira e sábado	Sem outlier

Fonte: Elaborado pelo autor.

perfil. O gráfico apresentado na Figura 39 exibe a média de acessos por dia da semana das notícias que possuem imagem de perfil.

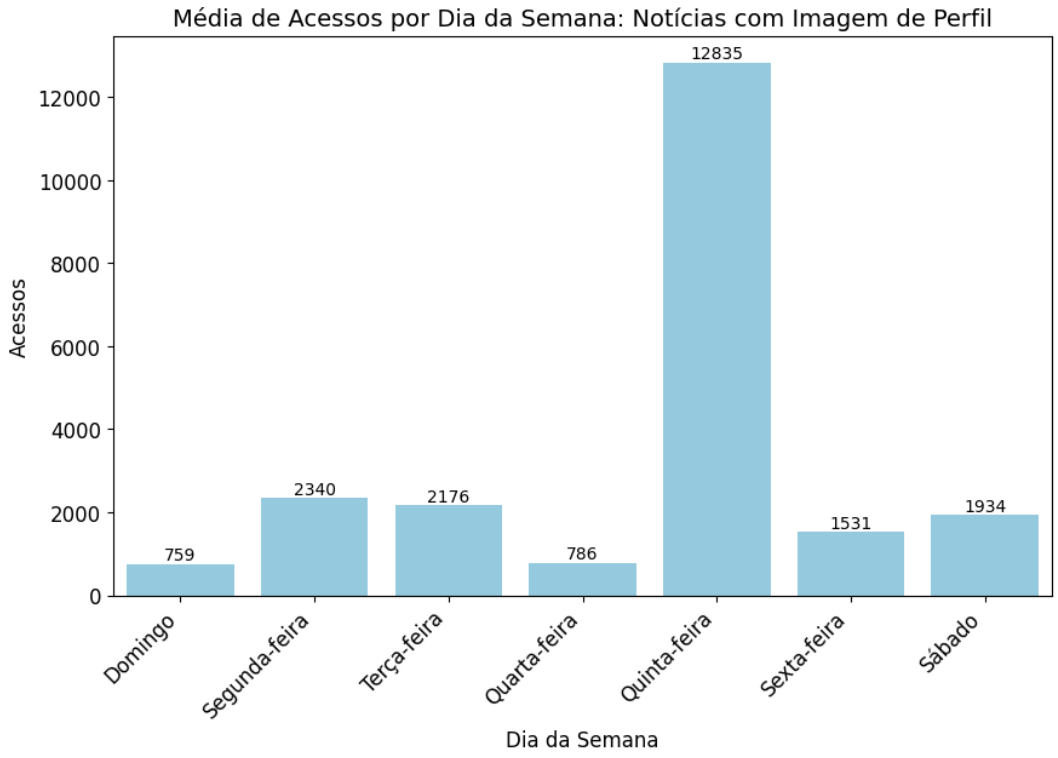

Figura 39: Média de acessos por dia da semana para as notícias com imagem de perfil.

Fonte: Elaborado pelo autor.

Como pode ser observado, quinta-feira apresenta um valor significativamente discrepante em relação aos demais dias da semana, podendo ser considerado um *outlier*. Dessa forma, o gráfico apresentado na Figura 40 mostra a mesma visualização, mas com a exclusão desse dia. Os dias com maiores médias de acessos para esse tipo de

notícia são: **segunda-feira**, com 2.340 acessos, seguida por terça-feira, com 2.176, e por sábado, com 1.934 acessos.

Assim, percebe-se uma certa correspondência entre as notícias com apenas 1 (uma) imagem, conforme a análise do gráfico da Figura 31: tanto segunda quanto terça-feira coincidem com o presente resultado. Na primeira análise, sexta-feira apresenta uma média de acessos ligeiramente superior à de sábado, que aparece logo em seguida; nesta, por sua vez, sábado apresenta maior média de acessos do que sexta-feira.

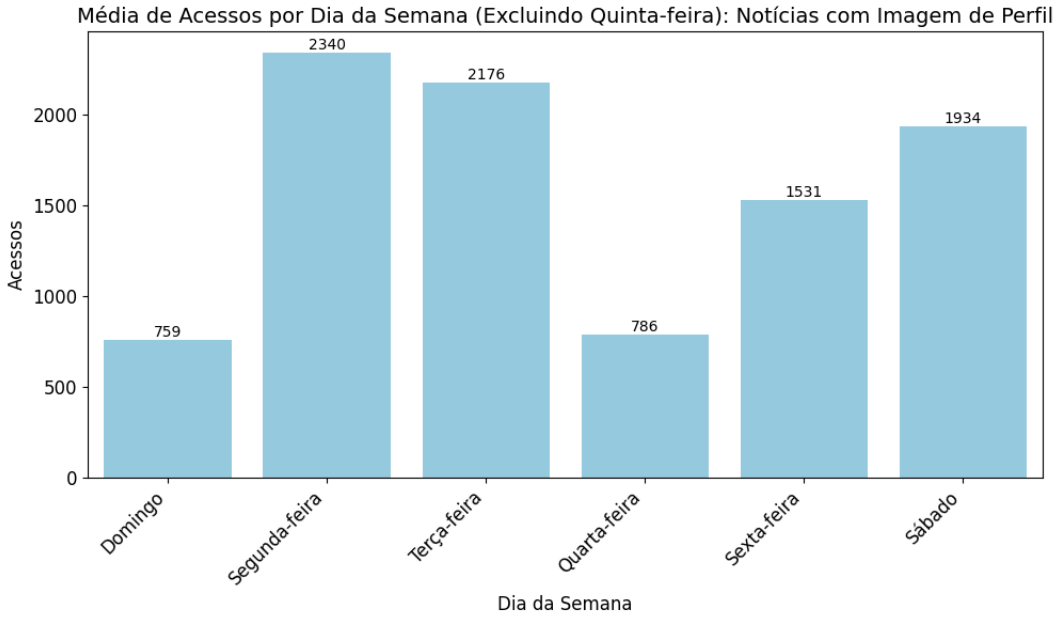

Figura 40: Média de acessos por dia da semana para as notícias com imagem de perfil, excluindo *outliers*.

Fonte: Elaborado pelo autor.

Da mesma forma, prosseguiu-se para a análise do grupo complementar ao anterior, isto é, das notícias **sem imagem de perfil**. Conforme apresentado no gráfico da Figura 41, os dias com maior média de acessos para esse tipo de notícia foram: **terça-feira**, com 7 acessos, seguida por segunda-feira e quinta-feira, ambas com 4 acessos, e pelos demais dias da semana, todos com 2 acessos. Assim, observa-se que o **2º** e o **3º lugar** resultaram em empate entre dias da semana.

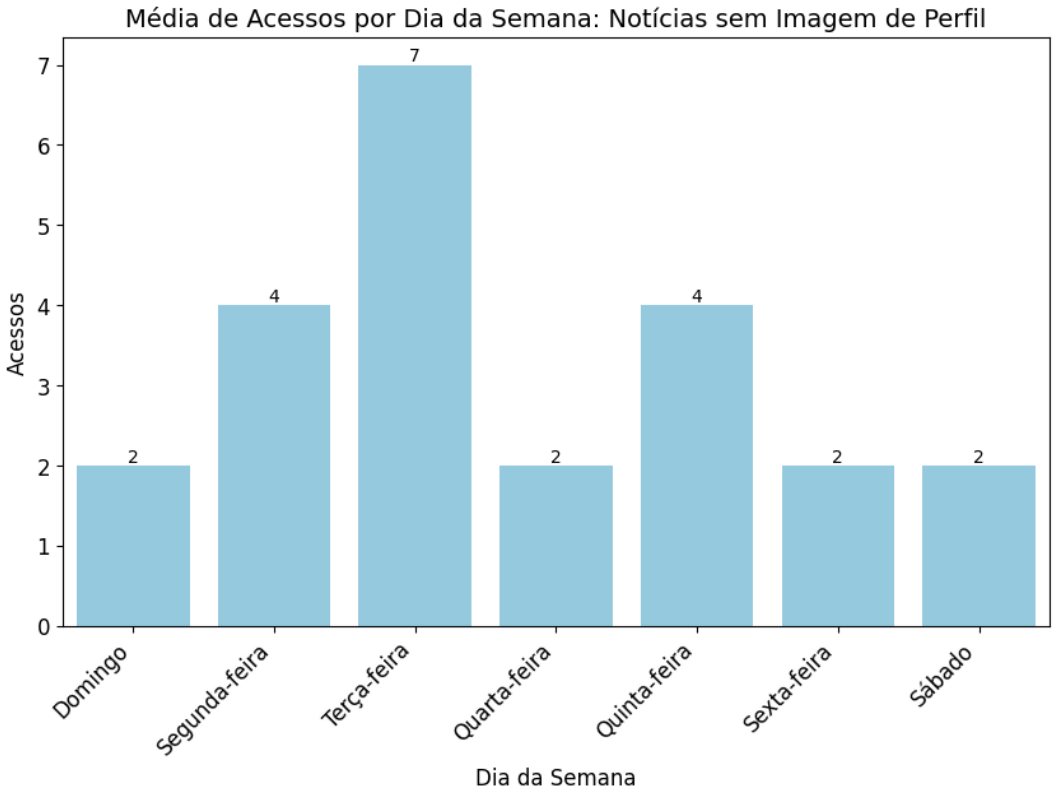

Figura 41: Média de acessos por dia da semana para as notícias sem imagem de perfil.

Fonte: Elaborado pelo autor.

Para finalizar essa análise, o gráfico apresentado na Figura 42 evidencia de forma clara a variação na quantidade de acessos ao longo do período de coleta dos dados das notícias no site do IF Goiano. Observa-se que, durante todo o período analisado, as notícias **com imagem de perfil** não apenas representam a maioria, como também predominam em termos de acessos, consolidando-se como o tipo de publicação mais atrativo para os usuários.

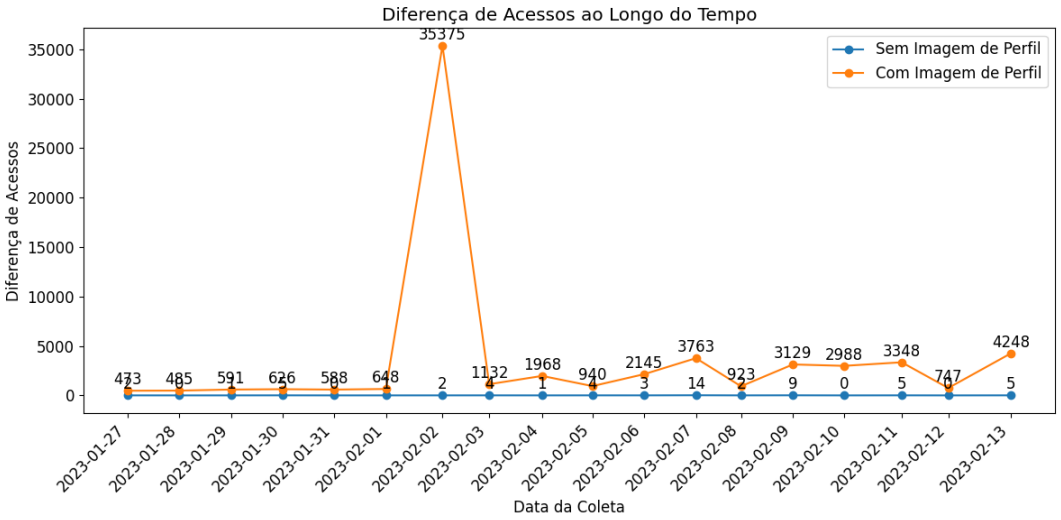

Figura 42: Diferença de acessos entre as notícias com e sem imagem de perfil ao longo do período de coleta.

Fonte: Elaborado pelo autor.

7.4 ANÁLISE DE SENTIMENTO SOBRE AS NOTÍCIAS

Partindo para a análise de sentimento aplicada ao conteúdo textual das notícias, agrupadas por assunto principal, concluiu-se que, conforme ilustrado no gráfico da Figura 43, embora as notícias sobre **Campanhas** e **Ações de Extensão** apresentem uma linguagem predominantemente positiva em comparação com os demais temas, elas não figuram entre as mais acessadas no site. Esse resultado indica que o uso de uma linguagem com sentimento positivo, por si só, ainda não é suficiente para despertar o interesse do leitor a ponto de motivá-lo a clicar na notícia.

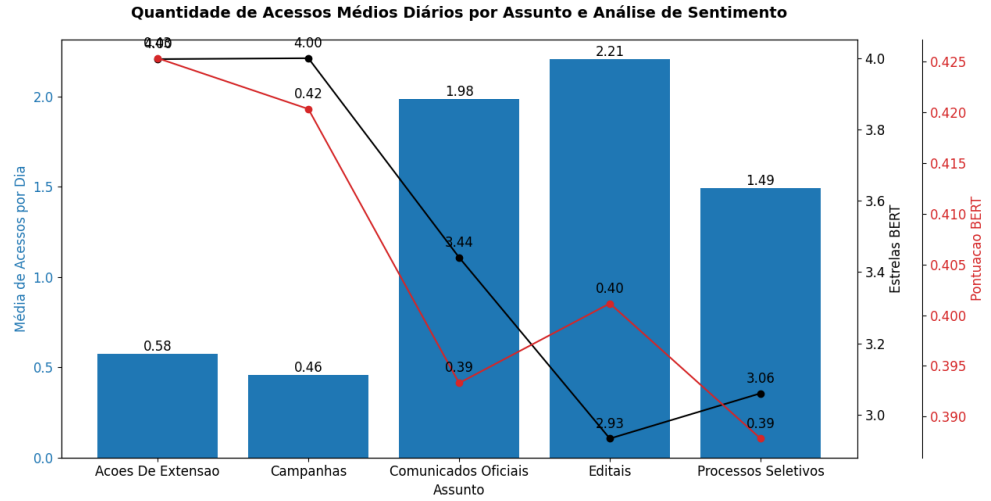

Figura 43: Análise de sentimento das notícias classificadas pelo tema, expressa em quantidade de estrelas e pontuação BERT.

Fonte: Elaborado pelo autor.

Com base no modelo para análise de sentimento utilizado, a quantidade de estrelas ou *label* (linha preta no gráfico) representa o sentimento predominante do texto, isto é, em uma escala de 1 a 5 estrelas, onde 1 corresponde a um sentimento muito negativo e 5 a muito positivo. Já a pontuação (linha vermelha no gráfico) expressa o grau de confiança do modelo em relação à classificação atribuída, indicando o quão seguro o modelo está sobre o sentimento identificado.

Por fim, após a sumarização do corpo das notícias e a subsequente comparação da análise de sentimento entre o texto original e o texto gerado pelo modelo BART, verificou-se que as descrições produzidas pelo modelo apresentaram desempenho superior na avaliação do modelo BERT em relação às descrições humanas.

Isso pode ser observado no gráfico da Figura 44, em que o texto gerado pelo BART obteve média de **3,74 estrelas**, frente a 3,31 do texto produzido pelo(a) jornalista, e no gráfico da Figura 45, em que o texto gerado pelo BART alcançou média de **0,40 na pontuação de confiança**, ligeiramente superior à média de 0,39 atribuída ao texto humano.

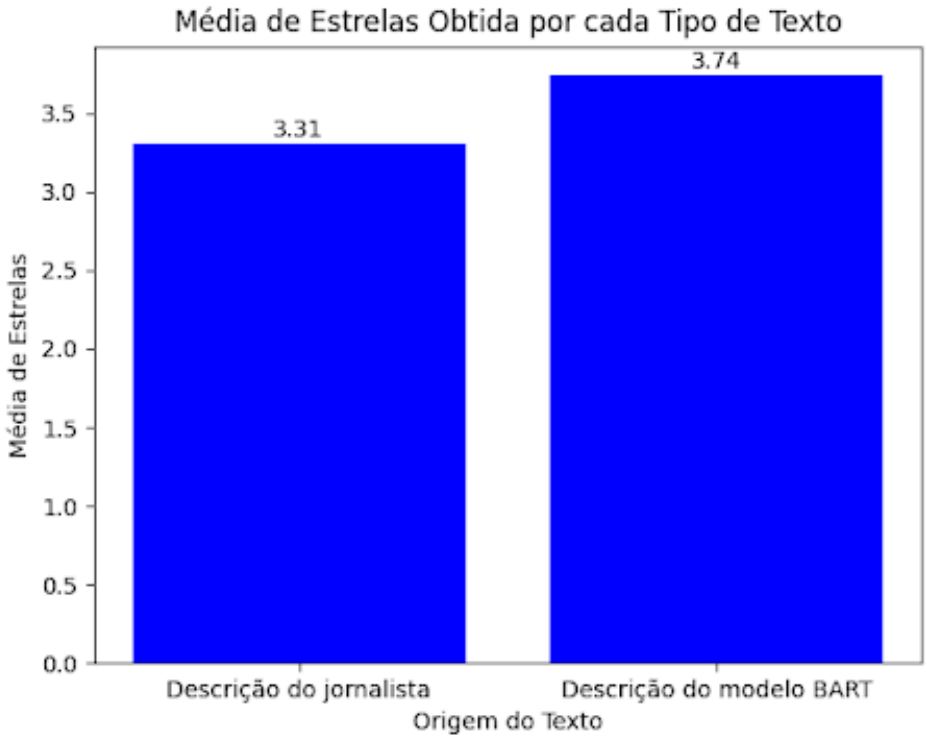

Figura 44: Comparação entre a média da quantidade de estrelas da descrição do jornalista e da descrição do modelo BART.

Fonte: Elaborado pelo autor.

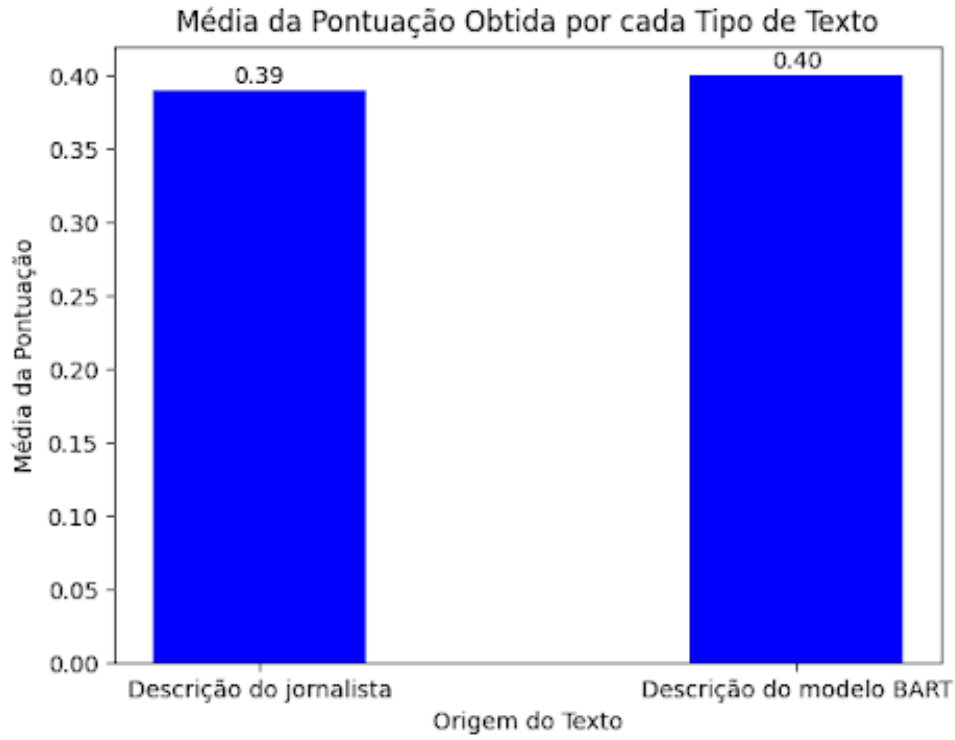

Figura 45: Comparação entre a média da pontuação da descrição do jornalista e da descrição do modelo BART.

Fonte: Elaborado pelo autor.

CONCLUSÃO

O trabalho desenvolvido nesta pesquisa teve como objetivo identificar, de forma sucinta e objetiva, os padrões de sucesso das notícias publicadas no site do IF Goiano, considerando múltiplas dimensões analíticas: o tema central das publicações, a variação de acessos dos usuários ao longo dos dias da semana, a quantidade de imagens presentes no corpo das notícias, a presença de imagem de perfil e a análise de sentimento aplicada aos títulos e às descrições das matérias.

Verificou-se que as notícias com maior número de acessos estavam predominantemente relacionadas a editais e processos seletivos publicados na plataforma, concentrando-se, em sua maioria, nas segundas, terças e quintas-feiras, e apresentando, geralmente, uma imagem de perfil associada à matéria.

Tal comportamento pode ser explicado pelo elevado número de vagas ofertadas pelo IF Goiano, destinadas a estudantes e profissionais da educação, o que torna o site um canal de consulta recorrente para candidatos e ingressantes interessados em novas oportunidades.

O sentimento das notícias não se mostrou um fator determinante para o aumento do número de acessos, independentemente do tema abordado. Observou-se que textos com maiores índices de positividade, ou seja, com vocabulário mais otimista e estrutura linguística mais elaborada, em alguns casos, obtiveram menos acessos do que aqueles com linguagem mais simples. Dessa forma, pode-se inferir que uma notícia com imagem de perfil e menor formalidade linguística tende a atrair mais leitores do que uma redação mais refinada, porém carente de apelo visual ou midiático.

É importante ressaltar que a análise das notícias do site institucional do IF Goiano concentrou-se nas publicações veiculadas na página principal da plataforma, cuja gestão editorial é de responsabilidade do Jornalismo da Reitoria. Esse setor é responsável pela administração e atualização do conteúdo disponibilizado, sem privilegiar um campus específico em detrimento dos demais. Dessa forma, a página principal reúne notícias de interesse institucional amplo, contemplando tanto o IF Goiano como um todo quanto suas diferentes unidades.

Além do site institucional, buscou-se realizar a mesma coleta e análise das

publicações do IF Goiano nas redes sociais Facebook e X. Entretanto, encontraram-se dificuldades técnicas na implementação do robô de software responsável pela extração dos dados, sobretudo devido à alta dinamicidade das estruturas das páginas dessas plataformas, que sofrem atualizações constantes, exigindo revisões frequentes e, por vezes, a reescrita completa do código desenvolvido.

Além disso, essas plataformas apresentam certo grau de instabilidade e estão em constante atualização quanto às suas formas de acesso aos dados, implementando políticas internas de uso cada vez mais restritivas. Tais políticas frequentemente limitavam ou até impediam a execução do software de coleta sobre as plataformas, mesmo quando se tratava de dados públicos disponíveis na rede.

Portanto, é possível afirmar que a principal dificuldade concentrou-se na etapa de coleta dos dados das notícias e/ou publicações nessas redes, o que, caso tivesse sido superado, poderia ter proporcionado um panorama mais abrangente das mídias sociais do IF Goiano e uma análise mais aprofundada e consistente dos padrões de engajamento e alcance institucional.

Assim, um próximo passo essencial para a continuidade desta investigação seria incluir as principais redes sociais em que o IF Goiano mantém perfis ativos e realiza publicações periódicas. Essas plataformas podem oferecer um volume ainda maior de dados, possibilitando a elaboração de um plano de análise mais amplo e representativo do comportamento do público e da eficácia comunicativa das publicações institucionais.

No caso do site do IF Goiano, os pesquisadores deste trabalho ficaram limitados ao formato de dados de acesso disponibilizado pela própria plataforma, que consiste em um contador global, isto é, o total acumulado de visualizações ao longo de todo o ciclo de vida da notícia desde sua publicação. Em razão dessa limitação, diversas análises, como a variação diária de acessos e o cálculo da média de visualizações por período, precisaram ser realizadas manualmente, por meio da implementação de algoritmos específicos desenvolvidos para este fim.

Portanto, a utilização dessas redes sociais possibilitaria a obtenção de um volume maior e mais diversificado de dados, como, por exemplo, reações e curtidas dos usuários em relação ao conteúdo publicado. Além disso, os comentários poderiam ser empregados na aplicação de algoritmos de análise de sentimento, permitindo uma com-

preensão mais aprofundada das percepções e opiniões dos usuários acerca das notícias divulgadas.

Retornando ao dilema central, a dinamicidade e as constantes atualizações nas plataformas das redes sociais, uma abordagem moderna e eficaz para contornar essa limitação consiste na utilização de agentes de inteligência artificial capazes de acessar páginas web e extrair seus conteúdos de forma automática e em tempo real, como, por exemplo, o *Firecrawl*[13] e o *ScrapeGraphAI*[14]. Esses agentes empregam modelos de IA para identificar, interpretar e extrair os dados relevantes, armazenando-os em estruturas organizadas e prontas para análise posterior.

[13]O endereço oficial para acesso se encontra em: https://www.firecrawl.dev/

[14]É possível conhecer mais sobre a tecnologia acessando: https://scrapegraphai.com/

REFERÊNCIAS

ALGHAMDI, Rubayyi; ALFALQI, Khalid. A Survey of Topic Modeling in Text Mining. International Journal of Advanced Computer Science and Applications (IJACSA), v. 6, n. 1, 2015. Disponível em: http://dx.doi.org/10.14569/IJACSA.2015.060121. Acesso em: 26 nov. 2025.

ALVES, Leonardo Emerson André. Caracterização, evolução e identificação de padrões em notícias falsas: uma abordagem voltada à modelagem de tópicos. 202 f. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação) – Instituto de Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2023.

BARBOSA, V. S.; ARAÚJO, A. D.; ARAGÃO, C. O. Multimodalidade e multiletramentos: análise de atividades de leitura em meio digital. RBLA, Belo Horizonte, v. 16, n. 4, p. 623-650, 2016. Disponível em: http://dx.doi.org/10.1590/1984-639820169909. Acesso em: 22 nov. 2025.

CAVNAR, W. B.; TRENKLE, J. M. N-Gram-Based Text Categorization. Article, 2001. Disponível em: https://www.researchgate.net/publication/2375544. Acesso em: 23 nov. 2025.

COHEN, K. B.; HUNTER, L. Getting Started in Text Mining. PLoS Computational Biology, New York, v. 4, n. 1, p. e20, 25 jan. 2008. Disponível em: https://doi.org/10.1371/journal.pcbi.0040020. Acesso em: 22 nov. 2025.

COSTA, Eduardo Santos da. A influência das notícias veiculadas pelo G1 e do índice ESG na explicação do comportamento do mercado acionário brasileiro. 28 f. Trabalho de Conclusão de Curso (Graduação em Administração) – Centro de Ciências Humanas, Sociais e Agrárias, Universidade Federal da Paraíba, Bananeiras, 2023. Disponível em: https://repositorio.ufpb.br/jspui/handle/123456789/31431. Acesso em: 22 nov. 2025.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. Disponível em: https://aclanthology.org/N19-1423.pdf. Acesso em: 23 nov. 2025.

EL-SAPPAGH, S. H. A.; HENDAWI, A. M. A.; EL BASTAWISSY, A. H. A proposed model

for data warehouse ETL processes. Journal of King Saud University - Computer and Information Sciences, v. 23, n. 2, p. 91-104, 2011. Disponível em: https://doi.org/10.1016/j.jksuci.2011.05.005. Acesso em: 22 nov. 2025.

FIGUEIREDO, Elaine B.; CATINI, Rita de Cássia; MENDES, Leonardo Manoel. Mineração de Textos: Análise de Sentimento em Redes Sociais – Revisão Sistemática. In: WORKSHOP EM COMPUTAÇÃO DO FÓRUM (WCF), 5., 2018. Anais do WCF 5. [S.l.: s.n.], 2018. p. 24-29. Disponível em: https://www.cc.faccamp.br/anaisdowcf/edicoes_anteriores/wcf2018/arquivos/04/paper_04.pdf. Acesso em: 22 nov. 2025.

KHDER, M. A. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. Int. J. Advance Soft Compu. Appl, v. 13, n. 3, Nov. 2021. Disponível em: https://www.i-csrs.org/Volumes/ijasca/2021.3.11.pdf. Acesso em: 22 nov. 2025.

LEWIS, Mike; LIU, Yinhan; GOYAL, Naman; GHAZVININEJAD, Marjan; MOHAMED, Abdelrahman; LEVY, Omer; STOYANOV, Ves; ZETTLEMOYER, Luke. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. [S. l.]: Facebook AI, 2019. Disponível em: https://arxiv.org/abs/1910.13461. Acesso em: 23 nov. 2025.

LOTFI, Chaimaa; SRINIVASAN, Swetha; ERTZ, Myriam; LATROUS, Imen. Web Scraping Techniques and Applications: A Literature Review. In: PAL, Raju; SHUKLA, Praveen Kumar (Ed.). SCRS Conference Proceedings on Intelligent Systems. Índia: SCRS, 2022. p. 381-394. Disponível em: https://doi.org/10.52458/978-93-91842-08-6-38. Acesso em: 22 nov. 2025.

LV, Junyan; XU, Shiguo; LI, Yijie. Application Research of Embedded Database SQLite. In: INTERNATIONAL FORUM ON INFORMATION TECHNOLOGY AND APPLICATIONS, 2009. [S.l.]: IEEE, 2009. p. 539-543. Disponível em: https://ieeexplore.ieee.org/document/5231398. Acesso em: 27 nov. 2025.

MEJOVA, Yelena. Sentiment Analysis: An Overview: Comprehensive Exam Paper. 2009. 28 f. Documento (Comprehensive Exam Paper) – Computer Science Department, University of Iowa, Iowa, 2009.

MOHR, Guilherme Alan; SILVA, Gustavo Pinto da; BRANDÃO, Janaína Balk; LICHTNOW, Daniel. Construindo um Dataset Relacionado à Produção e Comercialização de Produtos da Hortifruticultura no Brasil. In: ESCOLA REGIONAL DE BANCO DE DADOS (ERBD),

19., 2024, Farroupilha/RS. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 162-165. Disponível em: https://doi.org/10.5753/erbd.2024.238839. Acesso em: 22 nov. 2025.

QAISER, S.; ALI, R. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications, v. 181, n. 1, July 2018. Disponível em: https://www.researchgate.net/publication/326425709. Acesso em: 23 nov. 2025.

SANTHANAM, Sivasurya; HECKING, Tobias; SCHREIBER, Andreas; WAGNER, Stefan. Bots in software engineering: a systematic mapping study. PeerJ Computer Science, v. 8, e866, 2022. Disponível em: https://doi.org/10.7717/peerj-cs.866. Acesso em: 22 nov. 2025.

SINGH, Pramod. Airflow. In: SINGH, Pramod. Learn PySpark: Build Python-based Machine Learning and Deep Learning Models. Berkeley, CA: Apress, 2019. p. 115-136. DOI: 10.1007/978-1-4842-4961-1_4. Disponível em https://link.springer.com/chapter/10.1007/978-1-4842-4961-1_4. Acesso em : 29 nov. 2025.

TAN, Ah-Hwee. Text Mining: The state of the art and the challenges. In: PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (PAKDD), 1999, Beijing. Proceedings of the PAKDD'99 Workshop on Knowledge Discovery from Advanced Databases. [S.l.: s.n.], 1999. p. 65-70. Disponível em: https://www.researchgate.net/publication/2471634_Text_Mining_The_state_of_the_art_and_the_challenges. Acesso em: 22 nov. 2025.

VASWANI, Ashish; JONES, Llion; SHAZEER, Noam; PARMAR, Niki; GOMEZ, Aidan N.; USZKOREIT, Jakob; KAISER, Łukasz; POLOSUKHIN, Illia. Attention Is All You Need. 2017. Artigo (arXiv:1706.03762). Disponível em: https://arxiv.org/abs/1706.03762. Acesso em: 23 nov. 2025.