PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

A REALIDADE EM CONSTANTE DEFORMIDADE: CIÊNCIA DE DADOS APLICADA À ANÁLISE DE FAKE NEWS NO BRASIL

GUSTAVO DIAS MARTINS

GOIÂNIA
2024

GUSTAVO DIAS MARTINS

A REALIDADE EM CONSTANTE DEFORMIDADE: CIÊNCIA DE DADOS APLICADA À ANÁLISE DE FAKE NEWS NO BRASIL

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos necessários à obtenção do título de Graduação em Engenharia de Computação.

Orientador: Prof. Me. André Luiz Alves

Banca Examinadora: Prof(a). Esp. Nágela Bitar Lôbo e Prof. Me. Fabricio Schlag

GOIÂNIA

2024

GUSTAVO DIAS MARTINS

A REALIDADE EM CONSTANTE DEFORMIDADE:
CIÊNCIA DE DADOS APLICADA À ANÁLISE DE FAKE NEWS NO BRASIL

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação,

Aprovado em 18 / 06 / 2024.

Orientador: Prof. Me. André Luiz Alves

Prof(a). Esp. Nágela Bitar Lôbo

Prof. Me. Fabricio Schlag

GOIÂNIA
2024

AGRADECIMENTOS

Primeiramente devo agradecer a Deus, por todos os aprendizados e por ser meu guia até, e principalmente, em meus piores momentos. Proporcionou-me saúde e calma para meus pensamentos, auxiliando na concretização deste trabalho. Refletir sobre sua realização me faz acreditar em minha melhor versão.

O meu eu e todo o meu percurso na faculdade, não seriam os mesmos sem a presença da minha família. Agradeço incondicionalmente a todo carinho, apoio e confiança oferecidos pela minha mãe, pai e minha querida irmã mais velha. Vocês sempre me inspiraram e me moldaram, ser nem que seja um pouco de vocês se mostra como um dos meus maiores objetivos.

Agradeço imensamente ao professor, mestre e orientador André Luiz Alves, esse que sempre esteve presente demonstrando todo seu esforço em transmitir seus conhecimentos, conselhos e auxílio, sua presença foi fundamental para a realização deste trabalho. Toda minha insegurança e receio foram transformados em motivação e inspiração, exibindo os maiores significados de ter um professor. Evidentemente também direciono meus agradecimentos a todo o corpo de professores da Puc Goiás, suas aulas e nossas conversas impactaram diretamente no meu crescimento pessoal e profissional.

Agracio imensamente a presença de todos meus amigos, esbanjo sorte em ter pessoas maravilhosas desde da minha infância e também por conhecer novas durante os últimos anos. A presença e o apoio de vocês são fundamentais, assim como o companheirismo de minha namorada que me auxilia e inspira profundamente.

Agradeço do fundo do meu coração ao meu psiquiatra, psicólogo e cardiologista, graças a vocês pude enxergar a beleza e a vontade de viver. Por fim, gostaria de agradecer a todas as obras que consumo, mesmo parecendo algo fútil ou superficial, poder se identificar e abraçar qualquer material artístico é algo motivador.

RESUMO

Conviver cercado de atividades que geram e necessitam de dados fortalece a vontade de explorar esse material bruto em busca de resolver ou compreender melhor certos fenômenos. Definir a crescente evolução tecnológica vai além de desbravar os inúmeros impactos positivos; também se faz natural observar alguns fenômenos que seguem o caminho oposto dessa evolução, sendo as fake news um sólido exemplo. O movimento de disseminação de informações falsas demonstrou ser duradouro; o recente período pandêmico demonstra pontualmente o quanto a problemática continua atual e maliciosa. Este trabalho visa aplicar conhecimentos em torno da ciência de dados e do processamento de linguagem natural em um contexto de fake news, buscando compreender e avaliar a fabricação desse fenômeno, além de levantar uma maneira para realizar a verificação de um texto noticiário. Para isso, serão utilizadas tanto APIs do IBGE (uma fonte concreta de dados verídicos), quanto a utilização do Google News, que disponibilizará o material fonte de estudo, as notícias.

Palavras-Chaves**:** Fake News. Desinformação. Notícias. Ciência de Dados. Processamento de Linguagem Natural. API.

ABSTRACT

Living surrounded by activities that generate and require data strengthens the desire to explore this raw material in order to better understand or solve certain phenomena. Defining the growing technological evolution goes beyond exploring the numerous positive impacts; it also becomes natural to observe some phenomena that follow the opposite path of this evolution, with fake news being a solid example. The spread of false information has proven to be enduring; the recent pandemic period punctually demonstrates how current and malicious the issue continues to be. This work aims to apply knowledge in the field of data science and natural language processing in the context of fake news, seeking to understand and evaluate the manufacturing of this phenomenon, as well as proposing a way to verify a news text. To achieve this, both IBGE APIs (a reliable source of factual data) and the use of Google News will be employed, providing the study's source material— the news.

Keywords**:** Fake News. Disinformation. News. Data Science. Natural Language Processing. API.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	Application Programming Interface
AWS	Amazon Web Services
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CIA	Central Intelligence Agency
EUA	Estados Unidos da América
HTML	Linguagem de Marcação de HiperTexto
IBGE	Instituto Brasileiro de Geografia e Estática
IPCA	Índice de Preços ao Consumidor Amplo
KDD	*Knowledge Discovery in Databases*
OMS	Organização Mundial de Saúde
PIB	Produto Interno Bruto
PLN	Processamento de Linguagem Natural
SCIELO	Scientific Electronic Library Online
SIDRA	Sistema IBGE de Recuperação Automática
TER	Tribunal Regional Eleitoral
UFMG	Universidade Federal de Minas Gerais
USP	Universidade de São Paulo

SUMÁRIO

1 INTRODUÇÃO

Gradativamente a geração atual está cada vez mais imersa em um mundo soterrado por dados. Atividades como o registro da localização de um usuário em seu celular, de forma recorrente durante o dia, ou até mesmo a coleta de dados em carros e casas inteligentes, sempre analisando e coletando hábitos de seus usuários, enaltecem pontualmente esse estilo de vida. Muito também se deve as funcionalidades da Internet no dia a dia, de uma maneira geral a mesma pode representar um grande diagrama de conhecimento contendo uma volumosa enciclopédia de referências cruzadas: partindo de informações mais fúteis como mídias com cunho de entretenimento (jogos, vídeo, os apelidados *memes,* etc.), chegando até mesmo em inúmeras estatísticas governamentais (GRUS, 2016).

Entrelaçadamente com a ideia de que inúmeras ações, seja elas específicas ou convencionais, geram uma quantidade de dados, se torna indispensável a discussão em cima da área de ciência de dados.

A ciência de dados é um conceito a qual unifica diversos conhecimentos (estatística, análise de dados, *machine learning, etc*) com a finalidade de "compreender e analisar fenômenos reais" com dados (HAYASHI, 1998). Simplificadamente é ela que descobre quais dados podem ser úteis, identificando formas eficazes de gerenciá-los transformando os mesmos em conhecimento (KAMPAKIS, 2020).

Naturalmente ao observar sua ampla aplicação, a ciência de dados se torna cada vez mais uma prática recorrente em busca da solução de problemas ao redor de um grande acúmulo de dados.

Seguindo a contramão dos avanços tecnológicos, a desinformação se tornou uma característica marcante dos últimos anos do século XXI, a explosão das novas tecnologias e das redes sociais tornaram-se ferramentas úteis para facilitar a disseminação de informações falsas. Termos como *fake news* se tornaram recorrentes no dialeto das mais diferentes faixas de idade da população global. O Dicionário *Cambridge* define *fake news* como histórias falsas que se parecem com notícias, veiculadas pela internet ou por outros meios de comunicação, possuem o objetivo de influenciar opiniões políticas ou podem em algumas ocasiões apresentarem um teor satírico. Tais "notícias" demonstram um fenômeno recente de um problema antigo e complexo, se fazendo importante trazer uma opinião de quem atua na área de comunicações. Um artigo disponível

no portal do Tribunal Regional Eleitoral (TRE) de Goiás trouxe a visão do professor Eugênio Bucci, professor titular da Escola de Comunicações e Artes da Universidade de São Paulo (USP), acerca da problemática o professor reafirma pontos já levantados, mas também traz uma nova distinção de conceitos. Segundo o professor, as *fake news* seriam:

> "Um tipo historicamente datado de mentira. São uma criação do século XXI, que frauda a forma notícia a partir das plataformas sociais e das tecnologias digitais que favorecem a difusão massiva de enunciados" (TER, 2023).

Além dessa definição o docente faz uma reflexão em torno da diferença entre desinformação e *fake news,* em suas palavras "a desinformação é o feito geral da disseminação de *fake news* e de outros recursos para enganar ou manipular pessoas ou públicos com fins inescrupulosos". Através da reflexão discorrida pelo professor fica exposto o perigo de uma possível falta de capacidade social em distinguir o que é fato e opinião, sendo um reflexo direto da era da desinformação (TRIBUNAL REGIONAL ELEITORAL DE GOIÁS, 2023).

No dia 11 de março de 2020 a Organização Mundial de Saúde (OMS) declarava estado de pandemia para o COVID 19, doença proveniente do novo coronavírus Sars-Cov-2, situação essa que prevaleceu durante árduos e duradouros anos, foi apenas no dia cinco de maio de 2023 que foi decretado o fim do COVID-19 como uma emergência de saúde pública pelo o próprio chefe da OMS (NAÇÕES UNIDAS BRASIL, 2023).

O período foi marcado por questões econômicas, climáticas e políticas, entretanto não se pode relevar o impacto da disseminação de *fake news* durante esses anos, algo que também foi marcante perante a população brasileira.

Indiscutivelmente se torna de extrema periculosidade a disseminação de informações falsas em uma circunstância de alarme mundial em torno de um problema de saúde, ao passo que uma simples notícia pode influenciar um leitor a tomar cuidados indevidos, negligenciando assim sua própria vida afinal a pandemia resultou em mais de 696 mil mortes durante todo o seu intervalo (G1, 2023), número esse que comprova a alta letalidade da doença.

Apesar da pandemia ter sido superada nos tempos atuais não se pode afirmar o mesmo para as falsas noticiais. Uma pesquisa recente (2022), feita pela *Poynter Institute*, demonstra o quanto a problemática possui um grande escopo a partir do resultado de que, no Brasil, quatro em cada dez pessoas afirmaram receber notícias falsas todos os dias (GUIMARÃES e RODRIGUES, 2022).

Considerando as tecnologias atuais juntamente com todo o impacto que o fenômeno das *fake news* podem influenciar em diversos setores, de política até a área de saúde, se torna motivação suficiente refletir no estudo de uma maneira fundamentada e funcional para analisar notícias em busca de possíveis distorções de dados científicos.

O projeto visa descobrir uma maneira de aplicar os conceitos e métodos da ciência de dados em textos noticiários, analisando sua síntese e distorções. Com isso o trabalho buscará auxiliar no combate a disseminação de *fake news* podendo criar um ambiente mais seguro para pesquisas, preservando a relevância e notoriedade dos estudos levantados por pesquisadores.

Em sua estruturação, logo após a introdução, o capítulo 2 será responsável em destrinchar inúmeros conceitos necessários para a realização do projeto. O referencial teórico irá abranger assuntos como: conceitos acerca de dados, ciência de dados, análise de dados, Processamento de Linguagem Natural, população das fake news, etc.

Com os principais conceitos estabelecidos o capítulo 3 demonstrará os métodos e materiais necessários para a concretização da pesquisa, sendo apresentado o detalhamento de cada etapa.

O capítulo 4 visará destrinchar a aplicação do estudo no ambiente definido, sendo assim será responsável em detalhar os devidos resultados provenientes da aplicação da metodologia definida.

Por fim, o capítulo posterior, e último, é dedicado à conclusão. No mesmo será levantado as dificuldades enfrentadas durante a realização do trabalho, as decorrências dos resultados encontrados, além de levantar possíveis melhorias para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo demonstra certos conceitos relevantes para discussão e idealização do projeto, como: conceitos acerca de dados, ciência de dados, análise de dados, linguagem natural, etc.

2.1 Dados, Informação e Conhecimento

Comumente os conceitos de dados, informações e conhecimento acabam sendo interpretados de maneira equivocada, tornando relevante o esclarecimento pontual de suas diferenças.

Davenport e Prusak (1998) definem dados sendo observações acerca do estado do mundo, tendo a observação desses fatos brutos realizadas por pessoas ou uma determinada tecnologia apropriada. Segundo *Statistics Canada* dados são fatos, observações, números ou registros que podem assumir diversas formas de som, texto, imagem ou até mesmo medidas físicas, gerando conclusões acerca de sua coleta e processamento. Vídeos, filmes, cartas, extratos de contas, portais de notícias, registros feitos em um *smartwatch* são alguns dos exemplos possíveis para demonstrar a natureza dos dados. Enxergar sua abundância no cotidiano da população resulta em um senso de que "*Data is the new oil"* (dados são o novo petróleo), a partir do momento que sua exploração vem trazendo inúmeras vantagens e riquezas para o meio do emprenhadoríssimo (NETO, 2018).

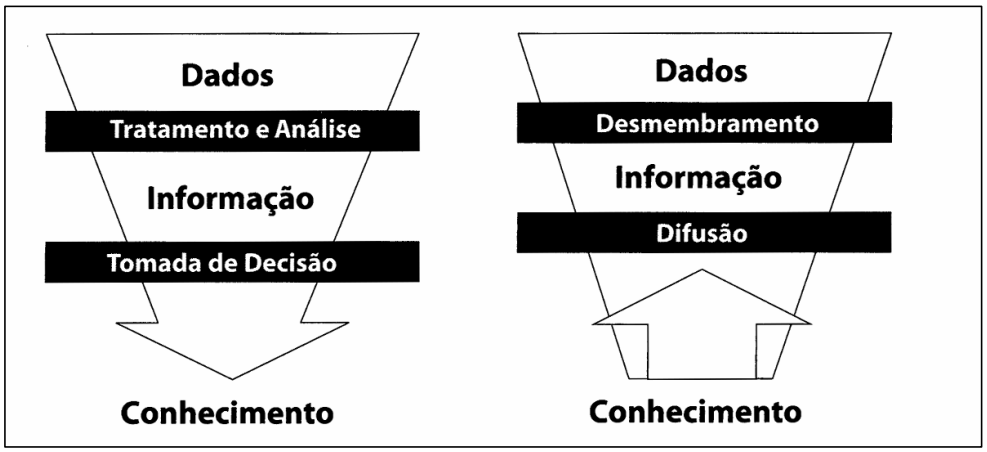
A informação é definida, por Peter Drucker, como "dados dotados de relevância e propósito". Essa atribuição acontece de forma natural pela própria interferência humana tornando complexo o tratamento das mesmas, pois as informações exigem análises e possuem uma certa dificuldade para a transferência com total fidelidade. Em suma a informação gera conhecimento a partir dos dados (DAVENPORT e PRUSAK, 1998).

Referindo-se a conhecimento, os mesmos autores especificam como sendo uma informação de maior valor sendo assim a mais complicada de gerenciar. O fato de ser valiosa gira em torno principalmente pela a forma que a própria é constituída: alguém se dispôs a refletir, conceder um contexto à informação, uma interpretação, um significado próprio. Intrinsicamente, o conhecimento é sempre agregado com a sabedoria de quem o observou, sendo

implícito a síntese de múltiplas fontes de informação (DAVENPORT e PRUSAK, 1998).

A Figura 1 demonstra de uma forma pratica a correlação dos conceitos descritos.

Figura 1 - Hierarquia dados, informação e conhecimento.



Fonte: Fávero e Belfiore (2017)

2.1.1 Tipos de dados

Refletir sobre a classificação que um dado pode receber demonstra uma peça fundamental para futuros estudos e análises em cima do mesmo, a partir do momento que as técnicas para realizar tais estudos dependem das características inerentes a essa classificação. Os dados podem ser definidos como estruturados e não estruturados (SHAH, 2020).

Dados estruturados demonstram uma alta organização das informações gerando uma facilidade em sua inclusão em um banco de dados e pesquisas. Enquanto os não estruturados partem do literal oposto, sendo assim desprovidos de estruturas subjacentes, além do fato dos valores diferentes desse tipo de dado não serem rotulados, algo que ocorre de forma contrária nos dados estruturados (SHAH, 2020).

2.2 Ciência De Dados

Frank Lo define a ciência de dados como sendo uma mistura multidisciplinar de inferência de dados, englobando o desenvolvimento de algoritmos e ferramentas para resolver problemas complexos no âmbito analítico

(2021). Chirag Shah em seu livro "*A hands-on introduction to data Science*" descreve a ciência como sendo um campo teórico e prático que em sua síntese engloba a coleta, armazenamento e processamento de dados resultando na obtenção de *insights* relevantes para um determinado contexto, um problema ou fenômeno. Dados esses gerados por humanos ou máquinas em diferentes formatos, como descrito em tópicos anteriores.

A Figura 2, traduzida e descrita no artigo de Rautenberg e Do Carmo, estampa a reunião de diferentes áreas em torno da prática de estudos em torno da ciência de dados.

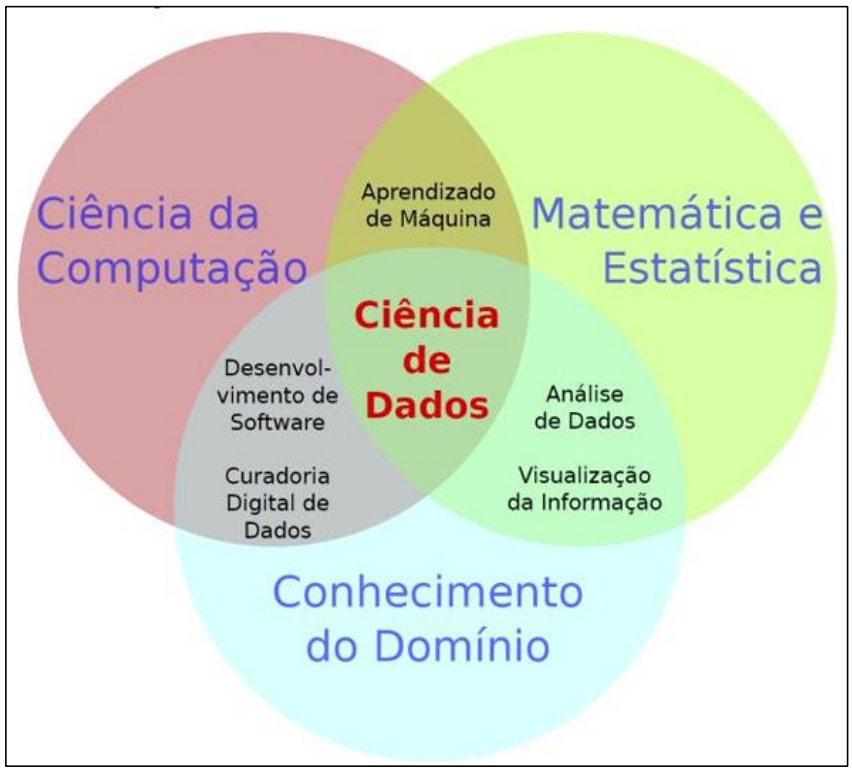
A ciência da computação se mostra uma habilidade necessária a partir do momento em que o armazenamento, manipulação e transmissão de dados são realizadas por computadores. Sendo uma área essencial para o Desenvolvimento de Software, Curadoria Digital e a implementação de *Machine Learning* (Aprendizagem de Máquina).

A matemática e a estatística são parte fundamental da análise de dados, para essa interpretação se torna inevitável a visualização da informação (o uso de gráficos é de extrema relevância para o processo de análise).

O sucesso de uma solução de ciência de dados é fundamentado pela demonstração do conhecimento do domínio do problema, sendo utilizado de forma ampla para o processo de tomada de decisão (RAUTENBERG e DO CARMO, 2019).

Já para Stylianos Kampakis (2020), existem três campos principais quando o assunto gira em torno de ciência de dados: A inteligência artificial, a qual se torna responsável em replicar uma máquina com inteligência suficiente para se aproximar de um cérebro humano valorizando um bom desempenho para aprendizagem, raciocínio lógico e autocorreção; o *machine learning* (aprendizado de máquina), termo a qual se refere à capacidade de um computador continuar melhorando e aprendendo com seus erros conseguindo ir além do escopo de sua programação; a estatística de certo modo é essencial, a mesma ajuda a desenvolver e estudar métodos para coleta, análise, interpretação e representação de dados.

Figura 2 - Ciência de dados e suas influências.



Rautenberg e Do Carmo (2019).

A influência da área se sustenta na falta limitação de sua aplicação, não sendo assim presa a uma única faceta da sociedade, tendo sua presença em praticamente em todos os lugares.

A área se viu fundamental para criação de inúmeras políticas públicas, de certo modo a ciência de dados ajuda governos e agências a obter *insights* a partir dos comportamentos dos cidadãos que influenciam a qualidade da vida pública: trânsito, transporte público, bem-estar tanto social quanto de comunidade, etc... (SHAH, 2021).

O próprio governo brasileiro (assim como o de outros países, como Canadá e Estados Unidos) disponibiliza cerca de mais de 12 mil conjuntos de dados abertos para uso, possibilitando assim o surgimento de projetos alvejando inúmeras funcionalidades.

A área de saúde destaca os benefícios do uso desse tipo de conhecimento a partir do momento que técnicas analíticas de ciência de dados possibilitam a análise e estudos acerca de inúmeros tipos de vírus, doenças, sintomas e diagnósticos. O setor financeiro é outro que se destaca pelo o bom uso dessas

ferramentas, muitas análises preditivas e de riscos são produtos da boa aplicação da ciência de dados (HUPDATA, 2020). Além dos exemplos citados, o marketing é outro meio que aproveita o potencial dos mecanismos da área descrita anteriormente, podendo ser visualizado em aplicações em torno de marketing direcionado, publicidade online e recomendações para venda cruzada (PROVOST e FAWCETT, 2016).

2.2.1 Big Data

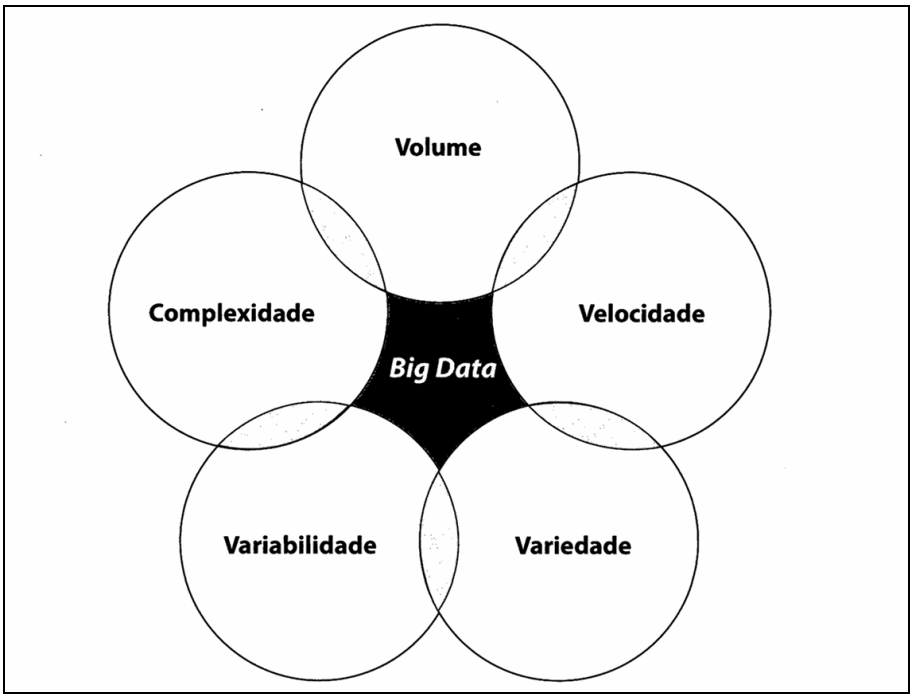
Além de ciência de dados, o termo *big data* se tronou bastante popular nos dias atuais. O termo se refere a conjuntos de dados com uma dimensão muito grande para os sistemas tradicionais de processamento, tornando-se assim maneiras novas parar processa-los. Em suma a tecnologia aplicada nesse contexto são de extrema utilidade para implementar técnicas de mineração de dados (PROVOST e FAWCETT, 2016).

Para Fávero e Belfiore (2017) o *Big Data* se fundamenta pela combinação de cinco dimensões de disponibilidade e geração de dados. Além do volume, como descrito anteriormente, outras características demonstram um grande destaque: a velocidade a qual os dados são disponibilizados, seja para tratamento ou estudo; a variedade entrelaçada pelas diferentes formas que dados podem ser acessados; a variabilidade dos dados; por último se destaca a complexidade dos dados.

A Figura 3 descreve de forma intuitiva a combinação a qual resulta o *Big Data*.

Para sustentar as dimensões descritas se torna primordial o constante aprimoramento de softwares profissionais, sustentado assim a principal razão para o crescente investimento em inúmeros setores, seja ele empresarial ou até mesmo acadêmico (FÁVERO e BELFIORE, 2017).

Figura 3 - *Big Data* e sua combinação.



Fonte: Fávero e Belfiore (2017).

2.2.2 Mineração de Dados

As bases de dados podem ser volumosas tornando o conhecimento um fator implícito, consequentemente se faz necessário uma busca detalhada com um processo analítico, automatizado e sistemático. A metáfora da palavra “mineração” se relaciona com a ideia de realizar uma busca detalhada (DA SILVA, PERES e BOSCARIOLI, 2016).

A mineração de dados (*Data Mining*) é definida como um processo semiautomático ou automático que explora de forma analítica bases grandiosas de dados, buscando encontrar padrões gerando um determinado conhecimento. O processo envolve a aplicação de técnicas capazes de receber um conjunto de fatos como entrada e devolve como saída um determinado padrão de comportamento (DA SILVA, PERES e BOSCARIOLI, 2016).

Em suma, a mineração está diretamente relacionada com o processo de Knowledge Discovery in Databases (KDD), ou em português Descoberta de Conhecimento em Bases de Dados. Descobertas de conhecimento a partir de bases de dados, visa encontrar principalmente padrões inerentes aos dados disponíveis. Resumidamente, o processo KDD se descreve como sendo um

processo analítico, sistemático e quando possível automatizado para a descoberta de determinado conhecimento. O processo inicialmente organiza as bases de dados; posteriormente ocorre o pré-processamento, o tratamento dos dados acontece aqui; a mineração de dados ocorre para no processo final os resultados serem validados, avaliados e formatados. A Figura 4 representa o esquema do processo (DA SILVA, PERES e BOSCARIOLI, 2016).

Figura 4 - O processo de KDD.

Fonte: Da Silva, Peres e Boscarioli (2016).

Quanto as tarefas do processo de mineração Fayyad *et al* (1996) apresentam uma taxonomia de dois níveis. Inicialmente as tarefas são divididas entre preditivas e descritivas: as preditivas utilizam valores descritivos para realizar a predição de valores desconhecidos, de certo modo tentam encontrar valores futuros; por outro lado, as tarefas descritivas apresentam o objetivo de descobrir padrões que descrever os dados (DA SILVA, PERES e BOSCARIOLI, 2016).

No segundo nível as tarefas são especializadas: nas descritivas, os autores incluem sumário, realizam agrupamentos, modelam dependências e incluem detecção de desvios; já para as preditivas, os autores “inserem classificação e regressão” (DA SILVA, PERES e BOSCARIOLI, 2016).

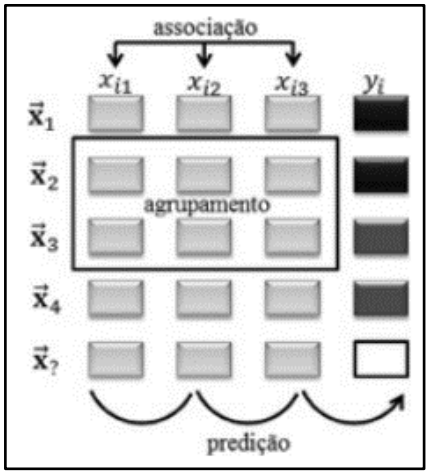
Da Silva et al (2016), apresentam uma visão mais abrangente reunindo 3 grandes tarefas quanto a mineração de dados: predição, agrupamento de dados e associações.

A tarefa em torno do agrupamento de dados de forma geral se resume na análise de conjuntos de dados pela presença das descrições dos mesmos, seu objetivo gira em torno de descobrir relações a partir da similaridade dos dados e a indicação de quais são semelhantes; tarefa de associação constituí pela busca

de ocorrências simultâneas e frequentes entre elementos de um determinado contexto, a principal meta da tarefa é encontrar padrões inesperados que eram desconhecidos (DA SILVA, PERES e BOSCARIOLI, 2016).

A Figura 5 representa a visão das tarefas especificadas pelos autores citados anteriormente.

Figura 5 - Tarefas de mineração de dados



Fonte: Da Silva, Peres e Boscarioli (2016).

2.3 Análise de Dados

Tratando de análise de dados, Santos a define como sendo a transformação de números (subsequente de técnicas e diferentes processos) com objetivo de otimizar uma tomada de decisão para determinado problema. Em suma, a análise possui como principal objetivo desvendar o significado atrás dos dados.

A análise identifica padrões, pode definir uma previsão de resultado futuro, compreende o desempenho passado, dentre outros auxílios para definir a tomada de decisão (SANTOS, 2016).

2.3.1 Categorias da análise de dados.

Pode-se destacar os seguintes tipos de análise: descritiva, preditiva, diagnóstica e exploratória.

A análise descritiva descreve de forma quantitativa as principais características de uma determinada coleção de dados. Mostra-se bastante

aplicada em grandes volumes de dados, sendo geralmente o primeiro tipo de análise realizado em uma base de dados (SHAH, 2021).

Análise preditiva demonstra a capacidade de prever um possível acontecimento a partir dos dados analisados, das tendências expostas e dos padrões já conhecidos (SHAH, 2021). Exemplo desse tipo de análise se destaca nos inúmeros algoritmos de previsão de preços.

Shah (2021) também define a análise diagnóstica como sendo a responsável em descobrir algo ou determinar o motivo de algo acontecer. Possuindo a correlação como técnica mais utilizada nesta análise: a técnica define-se como uma análise estatística a qual resulta na medição e descrição da força e da direção do relacionamento entre duas variáveis.

Sobre a última, define-se como sendo uma técnica presente em um contexto onde não se é conhecida a pergunta (hipótese), o analista deve descobrir respostas mesmo possuindo essa condição. A abordagem buscará encontrar relações que inicialmente eram desconhecidas (SHAH, 2021).

2.4 Processamento de Linguagem Natural

Bird *et al* (2009) define linguagem natural como sendo a linguagem convencional utilizada no cotidiano dos seres humanos, sendo intrinsicamente o contrário das linguagens artificias, ao passo que são linguagens de difícil definição de regras. O autor define Processamento de Linguagem Natural (PLN) como sendo qualquer tipo de manipulação computacional em torno da linguagem natural. Ainda sobre:

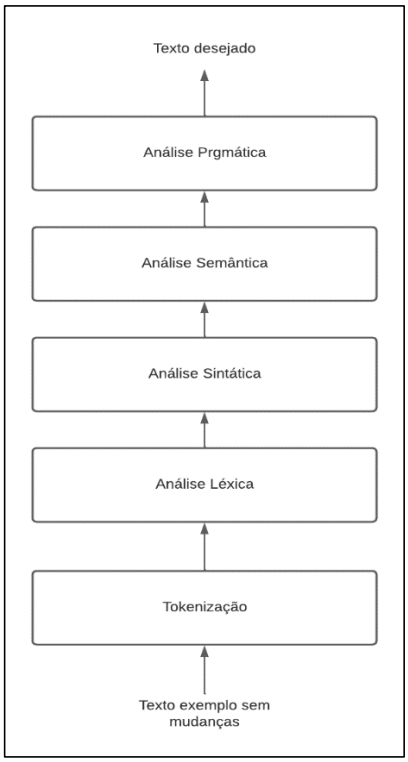
> Em um extremo, poderia ser tão simples quanto contar a frequência das palavras para comparar diferentes estilos de escrita. No outro extremo, a PLN envolve “compreender” declarações humanas completas, pelo menos ao ponto de ser capaz de dar respostas úteis a elas (BIRD *et al*, 2009, p. ix).

As tecnologias em torno do PLN vêm se tornando cada vez mais popular. A tradução automática de inúmeros textos para uma linguagem desejada ou a presença de reconhecimento de escrita manual em computadores e celulares, são exemplos desse aumento em sua difusão (BIRD *et al*, 2009).

Indurkhya e Damerau (2010) especifica que para o trabalho de análise de linguagem natural o processo deve ser dividido em alguns estágios: em primeiro lugar deve-se considerar que uma linguagem precisa ser analisada em termos de sua sintaxe; a partir desse processo, se vê possível se atentar para a uma análise semântica ou até mesmo do significado literal; por fim é feita uma análise pragmática a qual demonstrará o significado do trecho ou texto no contexto.

Entretanto, essa organização por muitas vezes serve apenas como um ponto de partida, se é considerado um processamento de textos reais, tornando se assim necessário um maior detalhamento dessas etapas. Dessa maneira, antes mesmo da análise sintática se faz necessário, respectivamente, a prática de tokenização e da análise lexical: é de extrema importância iniciar todo o processo realizando a tokenização e a segmentação de frases de um texto escolhido, isso se deve pela ausência de organização em uma linguagem natural; logo após, a análise léxica complementa a etapa anterior, agrupando todos os tokens descobertos (INDURKHYA e DAMERAU, 2010).

Figura 6 - Etapas de análise no Processamento de Linguagem Natural.



Fonte: Indurkhya e Damerau (2010).

A Figura 6 representa o esquema descrito em torno do Processamento de Linguagem Natural.

2.4.1 Pipelines treinados

O Processamento de Linguagem Natural, por muitas vezes utiliza-se de ferramentas para proporcionar uma maior facilidade para o futuro tratamento e análise do texto. O *Pipeline* de aprendizado de máquina define um conjunto de etapas interconectadas de processamento de dados e modelagem com intuito de automatizar, otimizar o processo de construção, padronizar, realizar o devido treinamento e implementar possíveis modelos de *machine learning. O* seu uso implica em uma maior intuição da complexidade do aprendizado de máquina, auxiliando cientistas de dados a desenvolver soluções precisas (IBM, 2024).

A biblioteca *Spacy*, focada em processamento de linguagem natural, demonstra um exemplo sólido da aplicação do método, já que permite a utilização de diferentes modelos (todos baseados em *pipelines* treinados) que impactam diretamente na manipulação de textos, abrangendo um grupo extenso de linguagem. Outro lado positivo se demonstra pela liberdade de alteração ou criação de um modelo próprio, seguindo a estrutura base já definida pela biblioteca (SPACY, 2024).

2.5 Application Programming

API ou *Application Programming Interface* (Interface de Programação de Aplicação) trata-se de mecanismos que permitem estabelecer uma comunicação entre dois componentes de software a partir de um conjunto de protocolos e definições. Um bom exemplo se faz a partir da aplicação para a exibição da previsão do tempo em um celular, destacando essa comunicação entre celular e API (AWS, s.d).

Outra definição importante gira em torno da API Web, a qual define-se como sendo “uma interface de processamento de aplicações entre um servidor da Web e um navegador da Web” de acordo com *Amazon Web Services* (AWS).

Independentemente de suas peculiaridades, as APIs se mostram de extrema relevância para uma possível aplicação em torno de grandes quantidades de dados, podendo gerar inúmeras análises relevantes. De certo, as próprias

acabam facilitando a obtenção de dados de uma forma segura, tendo em vista que inúmeras grandes instituições permitem um uso gratuito.

2.6 *Fake News* e sua popularização

Os perigos entorno das notícias falsas já se demonstra um assunto de bons anos atrás, em Réflexions d'Un Historien Sur les Fausses Nouvelles de la Guerre ("reflexões de um historiador sobre notícias falsas das guerras", Allia, 2012), ensaio feito por Marc Bloch e publicado originalmente em 1921, é demonstrado uma reflexão que continua de certo modo atual. O autor questiona a origem das falsas informações, de como as mesmas se propagam de forma rápida e exponencial, apontando o quanto essas podem mobilizar uma grande massa. Em uma de suas passagens Marc ressalta um fator importante para a circulação dessas notícias deturpadas, o próprio instiga a tese de que essas ideias falsas (em suas palavras, o erro) só se propagam, pois, há um "caldo de cultivo favorável na sociedade onde se expande", algo que pode ser visualizado desde dos primeiros grandes casos em torno de notícias falsa (ALTARES, 2018).

Demonstrando a longevidade da temática pode-se citar três grandes conflitos dos Estados Unidos (EUA) a qual se sustentaram em base de invenções: a guerra de Cuba (1898), a qual se beneficiou da manipulação dos jornais; o conflito do Vietnã, guerra que se sustentou a partir do incidente do golfo de Tonkin e pôr fim a invasão do Iraque em 2003, originada do forte discurso sobre as inexistentes armas de destruição em massa de Saddam Hussein (ALTARES, 2018). Sobre o último caso, o ex-analista da Agência Central de Inteligência (CIA), trabalhou durante 27 anos, Elizabeth Murray depôs em 2016 sobre sua experiência em torno do conflito do Iraque. Murray atuava fortemente na elaboração de relatórios que orientariam os EUA a partir da mídia iraquiana. Além de compreender melhor sua atuação, o analista pontuou de forma clara que Saddam Hussein não possuía relações com a Al-Quaeda, em suas palavras "Aquelas alegações eram pura ficção, assim como as alegações de que Saddam tinha armas de destruição em massa". (BRASIL DE FATO, 2016).

Entretanto foi em 2016, nas eleições presidenciais dos EUA entre Donald Trump e Hillary Clinton, que as *fake news* começaram a ganhar maior destaque. Uma análise realizada pelo *BuzzFeedNews* demonstrou que inúmeras noticiais falsas em torno das eleições presidenciais tiveram mais alcance no *Facebook* do

que de fato as verdadeiras noticiais veiculadas por mais de 19 grandes fontes de notícias (*New York Time*, *Washington Post* e *NBC News*), dentre das mais repercutidas se destaca: “Wikileaks confirma que Clinton vendeu armas para o Estado Islâmico” e também “Papa Francisco choca o mundo e apoia Donald Trump”. Também chama atenção o fato de que das 20 noticiais falsas, de melhor análise, apenas três não eram pró-Trump ou contra Hillary Clinton (G1, 2016). Inclusive, Trump foi uma figura marcante quando o assunto é divulgar (até mesmo se beneficiar) de *fake news*, algo que se estendeu até seus últimos momentos como presidente (2017-2021): o ex-presidente acusou o México de enviar criminosos para os Estados Unidos; afirmou que tanto Hilary Clinton quanto Barack Obama haviam criado o Estado Islâmico; acusou que o sistema de eleição de seu país (especificamente votos pelos correios) seriam roubados, causando um possível resultado fraudulento da eleição a qual perdeu (algo que se provou o contraio); fez inúmeras afirmações inverídicas sobre o COVID-19 (G1, 2016).

Em torno do assunto COVID-19, algumas narrativas podem ser destacadas: a falsa confirmação de que um estudo da Universidade de Harvad apontou como um método de prevenção do COVID-19 o uso da hidroxicloroquina, algo que se provou falso ao ponto que não houve comprovação sendo que o estudo não teve quaisquer relações com a universidade (COMPROVA, 2022); a afirmação em inúmeras mensagens de que a vacina para COVID-19 causava varíola dos macacos, desmentida por diversos profissionais, destacando-se Flávio da Fonseca virologista da Universidade Federal de Minas Gerais (UFMG) e presidente da Sociedade Brasileira de Virologia, a qual refutou a ligação da varíola com as vacinas (DOMINGOS, 2022); um vídeo circulou pelas mídias referente a lamentação de um homem pela morte de uma criança, afirmaram que a morte havia sido proveniente do efeito da vacina para COVID-19, mentira essa exposta pelo fato do contexto da imagem estar relacionada com um bombardeio na Síria em 2021 (DOMINGOS, 2022); outra invenção está na narrativa de que a variante Ômicron (uma das variantes do coronavírus) havia sido inventada para disfarçar efeitos colaterais das vacinas, o médico infectologista Leonardo Weissmann juntamente com a doutora em biociências e fisiopatologia pela Universidade Estadual de Maringá, Letícia Sarturi, desmentem cada ponto dessa falsa notícia em texto publicado pelo G1 (DOMINGOS, 2021).

2.6.1 *Fake News* e fatos reais

É das gêneses das apelidadas *fake news* distorcer dados já validados e pesquisados, em busca da construção de uma narrativa inverídica. Já alertava Roberto Olinto, presidente do Instituto Brasileiro de Geografia e Estática (IBGE), em 2018 sobre como essa divulgação de informações falsas impactam diretamente os órgãos estatísticos. O próprio afirma:

> "Volta e meia sai uma notícia que, associada a alguma questão econômica, social, [dá a entender que] o IBGE está envolvido nessa confusão". Toda essa grande disseminação pode chegar ao ponto de gerar dúvidas na população sobre estatísticas factíveis como Olinto cita "50% da população brasileira acha que o IPCA [Índice de Preços ao Consumidor Amplo] é errado, que dentro da sua casa a inflação não é aquela e não acredita na gente de forma nenhuma" (MALLES, 2018).

3 METODOLOGIA

Este capítulo inicialmente categoriza a tipologia da pesquisa em torno do projeto para depois ser demonstrado o passo a passo feito para alcançar os objetivos em torno da problemática.

3.1 Definindo Métodos

Esta pesquisa segundo sua natureza se trata de um resumo de assunto, pois a própria de certa forma busca sistematizar uma área de conhecimento, ou seja, a pesquisa busca organizar conhecimentos já conhecidos gerando uma discussão acerca do tema (WAZLAWICK, 2014). A pesquisa é classificada da seguinte forma, pois a mesma esquematiza e analisa conhecimentos e tecnologias já estabelecidas (conceitos em torno de *data science,* uso de APIs, estudos em torno do fenômeno das *fake news,* dentre outros).

Segundo seus objetivos a pesquisa é caracterizada como descritiva, principalmente pelo fato da mesma buscar obter dados mais consistentes da realidade proposta (WAZLAWICK, 2014). Para alcançar os objetivos será necessário o estudo de diferentes técnicas em volta da ciência de dados e processamento de linguagem natural, além de utilizar o uso de algoritmos para levantar dados a partir das notícias disponíveis pelo Google e estatísticas provenientes do IBGE.

Segundo seus procedimentos técnicos a pesquisa se define como sendo: bibliográfica, documental, experimental e quantitativa (WAZLAWICK, 2014).

Primeiramente para a realização do projeto, foi feito o estudo breve sobre definições e conceitos relacionados as áreas especificadas. Foi necessário a seleção de artigos de diversas fontes, dentre elas se destaca os periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), o SciElo, além do uso de livros. Caracterizando assim como uma pesquisa bibliográfica (WAZLAWICK, 2014). A pesquisa dos artigos girou em torno dos temas: ciência de dados, *fake news,* linguagem natural e pandemia COVID-19.

Classifica-se como documental ao passo que o projeto busca gerar informações e padrões em torno de dados encontrados em diferentes documentos (WAZLAWICK, 2014). O conjunto entre dados provenientes da API do IBGE e do governo brasileiro, são utilizados para estabelecer uma análise dos textos de noticiais, esses também provenientes de uma API.

Após a coleta de inúmeros dados em diferentes meios, se torna consequente a análise desse aspecto da realidade (*fake news* nas mídias nesse caso). A partir das variáveis selecionadas pode se destacar padrões, dependências, gerando conclusões acerca da análise feita em torno de diferentes notícias. Demonstrando ser também uma pesquisa experimental (WAZLAWICK, 2014).

Relacionado com os modelos expostos anteriormente, o método quantitativo também é destacado. Método esse definido pelo ato de mensurar variáveis, é a partir das mesmas que é capturado diversas evidências (MIGUEL, 2012). Todo o processo de organização dos dados obtidos em data frames e gráficos demonstram pontualmente essa ideia de quantizar variáveis selecionadas, gerando uma análise posterior em cima dessas representações.

3.2 Ambiente Python

Originada no fim da década de 80 por Guido Van Rossum, a linguagem de programação *Python* demonstra uma vasta variedade de aplicações além de proporcionar uma certa facilidade em seu uso. Considerando sua agilidade, praticidade e versatilidade (CARVALHO, 2023) se torna conveniente o seu uso para a resolução da problemática apontada.

Dentre das inúmeras ferramentas proporcionadas pela linguagem, uma de suas estruturas básicas impacta diretamente na resolução do projeto. Os dicionários apresentam uma estrutura semelhante de uma lista, entretanto sua diferença está no fato dos mesmos não serem indexados por inteiros. A estrutura de dados se mostra indexada por chaves (*Keys*), a qual podem ser de qualquer tipo imutável (PYTHON, 2024). Tal característica resulta em uma solução responsiva para o armazenamento de dados.

O exemplo a seguir, ilustra a diferença e a utilidade da estrutura:

- Dic_ex = {‘País’: ‘Brasil, ‘Cidade’: ‘Goiânia’, ‘Data’: ‘16/09/2021’}: o caso representa a criação de um dicionário possuindo chaves referentes ao país, cidade e data de acesso de algum usuário fictício. A estrutura permite uma maior facilidade tanto de legibilidade quanto de acesso ao um dado especifico;

- Para o caso de uma lista, considerando o cenário descrito, o resultado seria: List_ex = [‘Brasil’, ‘Goiânia’, ‘16/09/2021’].
- A requisição de uma data presente em ambas estruturas resultaria: a lista retornaria o valor da posição 2 (List_ex[2]) enquanto o dicionário retornaria o valor a partir da chave ‘Data’ (Dic_ex[‘Data’]).

Partindo da visão geral em torno da linguagem e da possibilidade de encontrar ferramentas úteis, foi selecionado o uso do ambiente proporcionado pelo *Pycharm Professional*, uma boa plataforma para diferentes aplicações a qual estabelece a possibilidade de programar utilizando *Jupyter Notebook* de uma forma otimizada. Resultando uma manipulação de dados otimizada pela divisão de células.

3.3 Seleção de Dados

Com o ambiente já estabelecido a atenção deve ser voltada para o levantamento dos dados necessários para a concretização da aplicação. Dados provenientes de pesquisas do IBGE e textos noticiários serão coletados e tratados.

3.3.1 Dados do IBGE

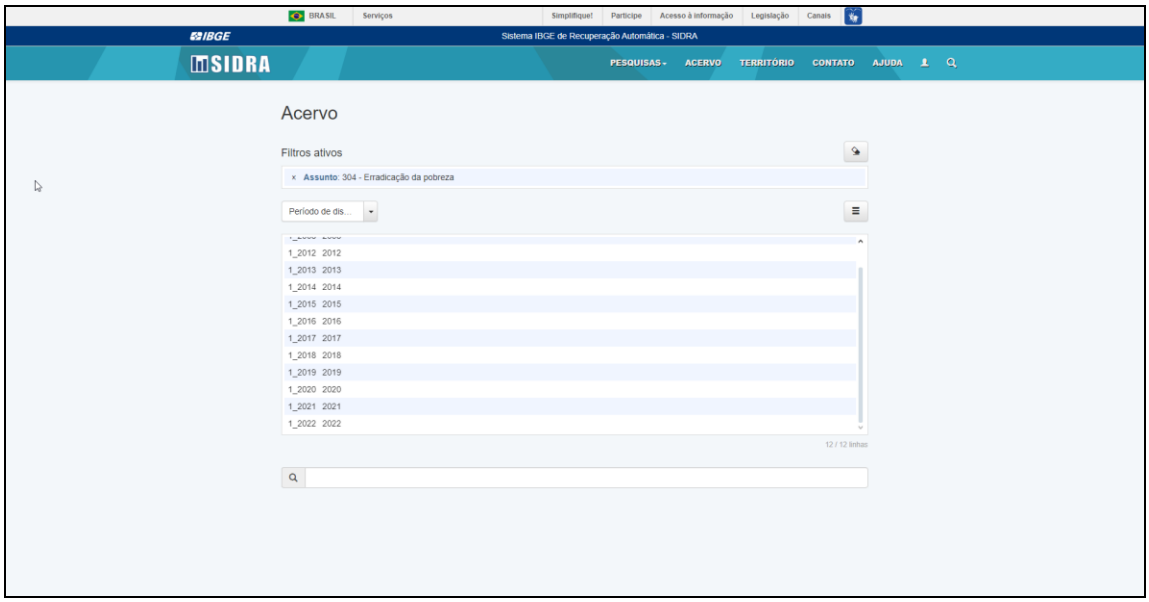
A verificação da veridicidade de um dado presente em um texto, é feita a partir da tentativa de consulta do mesmo nos dados pesquisados e estudados pelo IBGE. Sendo assim necessário o uso das APIs do IBGE descritas rapidamente em tópicos anteriores, a partir das mesmas a base de dados verídicos será construída.

A ideia base gira em torno de reunir uma grande quantidade de informações de setores socioeconômicos, a qual em vários momentos as pesquisas científicas são alvos de distorções. Dito isso, inicialmente é utilizado a API de agregados a qual retorna todos os agregados provenientes de determinados assuntos. A partir de cada agregado, pose-se estabelecer os seus metadados em torno dos parâmetros: período, assunto, classificação, periodicidade e nível.

A API utilizada abrange diversos assuntos de diversas áreas (são mais de 300 assuntos), entretanto não são todos que estão devidamente atualizados. Logo o primeiro critério para a seleção dos assuntos foi possuir dados entre o intervalo de 2020 a 2024. O site do Sistema IBGE de Recuperação Automática (SIDRA)

permite uma fácil verificação do período para cada assunto, a Figura 7 demonstra um exemplo.

Figura 7 - Verificando a periodicidade do assunto Erradicação da Pobreza.



Fonte: Sistema IBGE de Recuperação Automática (2024).

Visando cumprir o critério de tempo e a ideia de diminuir o escopo do projeto, os temas selecionados estão presentes no Quadro 1. Dessa maneira, dos diversos assuntos disponibilizados pela API do IBGE, os assuntos escolhidos para a produção do projeto são expostos no Quadro 1, exibindo seus respectivos Ids definidos pela API do IBGE.

Quadro 1 - Assuntos escolhidos da API do IBGE.

ID	Assunto
304	Erradicação da pobreza
320	Furto e roubo
46	Redução das desigualdades
67	Saúde
274	Meio Ambiente

Fonte: Sistema IBGE de Recuperação (2024).

Os agregados são obtidos a partir de *urls* definidos pela própria API, dessa maneira a função “gerar_agregados” possibilita a geração do *link* correspondente ao ID do assunto alvo, fazendo assim necessário a utilização da biblioteca “*requests*” (biblioteca especializadas em realizar requisições *web*). A partir do *link* gerado, é salvo em uma lista, própria para cada assunto, todos os agregados disponíveis do tema (função “agregados”), sendo cada item um dicionário com diferentes chaves. Cada agregado possui um Id próprio para seus respectivos metadados, que também são obtidos por um *url* próprio, fazendo assim necessário gerar um novo *link* (função “gerar_metadados”) e armazenar os respectivos resultados em uma nova lista (função “lista_metadados”). A Figura 8 demonstra a definição das funções enquanto a 9 revela o uso prático das mesmas, assim como a utilização da biblioteca “pandas” para a criação de *DataFrames* (dados organizados em formato de tabela). A biblioteca demonstra ser fundamental para uma boa estruturação em torno de uma grande quantidade de dados, alcançar uma boa visualização torna mais dinâmico o tratamento dos dados.

Figura 8 - Definição das funções de agregados e metadados.

```python
def gerar_agregados(id):
    link = f"https://servicodados.ibge.gov.br/api/v3/agregados?assunto={id}"
    return requests.get(link).json()

def agregados(lista):
    lista_retornar = []
    for i in lista[0]['agregados']:
        lista_retornar.append(i)

    return lista_retornar

def metadados(id):
    link = f"https://servicodados.ibge.gov.br/api/v3/agregados/{id}/metadados"
    requisicao = requests.get(link).json()
    return requisicao

def lista_metadados(lista):
    lista_meta = []
    for i in lista:
        aux = metadados(i['id'])
        lista_meta.append(aux)
    return lista_meta
```

Fonte: Elaborado pelo autor (2024).

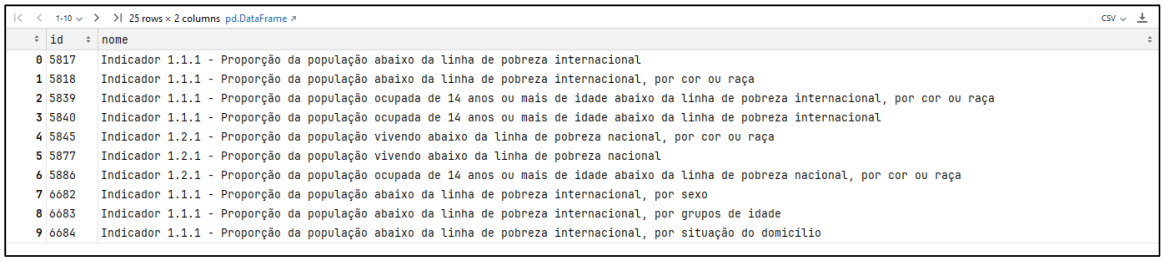
Figura 9 - Utilizando as funções para o assunto Erradicação da pobreza.

```python
# Relevante 304 Erradicação da pobreza
pobreza_agregados = gerar_agregados(304)
list_agre_pobreza = agregados(pobreza_agregados)
data_pobreza = pd.DataFrame(list_agre_pobreza)

# Tratando Metadados 304 Erradicação da pobreza
pobreza_meta = lista_metadados(list_agre_pobreza)
pobreza_classi = classificacoes_metadados(pobreza_meta)
```

Fonte: Elaborado pelo autor (2024).

A Figura 10 exibe um exemplo da disposição em *DataFrame* dos agregados para o tema “Erradicação da pobreza”, evidenciando a importância da organização.

Figura 10 - *DataFrame* dos agregados da Erradicação da pobreza.



1-10 | 25 rows × 2 columns | pd.DataFrame | CSV

	id	nome
0	5817	Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional
1	5818	Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional, por cor ou raça
2	5839	Indicador 1.1.1 - Proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional, por cor ou raça
3	5840	Indicador 1.1.1 - Proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional
4	5845	Indicador 1.2.1 - Proporção da população vivendo abaixo da linha de pobreza nacional, por cor ou raça
5	5877	Indicador 1.2.1 - Proporção da população vivendo abaixo da linha de pobreza nacional
6	5886	Indicador 1.2.1 - Proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza nacional, por cor ou raça
7	6682	Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional, por sexo
8	6683	Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional, por grupos de idade
9	6684	Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional, por situação do domicílio

Fonte: Elaborado pelo autor (2024).

Com os metadados a disposição, a última etapa para a organização dos dados, provenientes do assunto no caso erradicação da pobreza, se torna mais próxima. Antes mesmo de ser atingida, se vê necessário armazenar as classificações e categorias de cada metadado de uma maneira prática. O tratamento é relevante ao passo que os mesmos posteriormente definirão os tipos de dados (e seus respectivos valores), e como serão buscados e armazenados através da API do SIDRA, implementada pela biblioteca “*sidrapy*”. A Figura 9 também aponta o uso prático da função “classificações_metadados”, responsável pelo tratamento das classificações, tendo sua estruturação descrita na Figura 11.

Figura 11 - Funcionamento função do tratamento de classificações.

```
def classificacoes_metadados(lista_metadados):
    lista_classificacoes_metadados = []
    for i in lista_metadados:
        strlista = []
        dicionario = {}
        period_freq = i['periodicidade']['frequencia']
        period_fim = i['periodicidade']['fim']

        if period_freq == 'anual':
            if period_fim >= 2020:
                nivel_ter = i['nivelTerritorial']['Administrativo']
                dicionario['Id'] = f"{i['id']}"
                dicionario['Nome'] = f"{i['nome']}"
                for j in i['classificacoes']:
                    for k in j['categorias']:
                        strlista.append(str(k['id']))
                    clas_tratada = ','.join(strlista)
                    classificacao = f"{str(j['id'])}/{clas_tratada}"
                    dicionario['Classi_Str'] = classificacao
                    dicionario['Nível_Territorial'] = nivel_ter
                    dicionario['Periodo'] = period_freq
                    dicionario['Fim'] = period_fim
                    lista_classificacoes_metadados.append(dicionario)

        elif period_freq == 'mensal':
            if period_fim >= 202001:
                nivel_ter = i['nivelTerritorial']['Administrativo']
                dicionario['Id'] = f"{i['id']}"
                dicionario['Nome'] = f"{i['nome']}"
                for j in i['classificacoes']:
                    for k in j['categorias']:
                        strlista.append(str(k['id']))
                    clas_tratada = ','.join(strlista)
                    classificacao = f"{str(j['id'])}/{clas_tratada}"
                    dicionario['Classi_Str'] = classificacao
                    dicionario['Nível_Territorial'] = nivel_ter
                    dicionario['Periodo'] = period_freq
                    lista_classificacoes_metadados.append(dicionario)
    return lista_classificacoes_metadados
```

Fonte: Elaborado pelo autor (2024).

O tratamento de cada item da lista de metadados passa por uma simples filtragem de acordo com sua periodicidade: caso seja definido como anual, é verificado se o mesmo possui como último período, algum dentro da faixa estabelecida anteriormente (2020 – 2024). Sendo verdadeiro, cada categoria é armazenada em uma única sentença, separando cada categoria por vírgula, para posteriormente armazenar como um novo item de uma nova lista a classificação já tratada (formada por um Id da classificação, seguida de uma barra (“/”) e pela sentença já criada); caso a frequência do período seja definida pelo tipo mensal, torna necessário verificar se o último período está presente no intervalo de tempo, a diferença para o caso anterior está no fato dessa situação verificar o código relacionado ao período e não necessariamente o mesmo. O processo ocorre por ser mais ágil e intuitivo realizar comparações numéricas ao invés de verificar o trimestre e ano por extenso. Logo após, os mesmos passos são repetidos para a criação do novo item de uma nova lista, a qual é retornada juntamente com seus novos valores e com antigos que definem pontualmente o tipo daquele metadado (como nome e Id).

A Figura 12 exemplifica o modelo de valores retornados pelo tratamento da função “classificações_metadados”.

Figura 12 - Exemplo de um item da lista retornada pela função.

```
{'Id': '5818',
 'Nome': 'Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional, por cor ou raça',
 'Classi_Str': '86/0,95251,2776,119525',
 'Nível_Territorial': ['N1'],
 'Periodo': 'anual',
 'Fim': 2022}
```

Fonte: Elaborado pelo autor (2024).

Os itens já tratados são passados para a função “gerar_dataframes_sidra”, a qual retornará todos os itens com seus respectivos *DataFrames*. A biblioteca “*sidrapy*” possibilita a chamada/criação de uma tabela a partir de códigos em *Python*, sendo necessário a passagem de determinados parâmetros, que já se encontram tratados em cada item da lista, sendo eles: Id do tema, nível territorial (define em quais regiões a pesquisa engloba, podendo ser o país inteiro ou alguma região específica), o período e todas as classificações comportadas. Estes

serão a principal ferramenta para a verificação de um dado, todas as consultas serão feitas a partir das tabelas criadas e armazenadas para cada assunto.

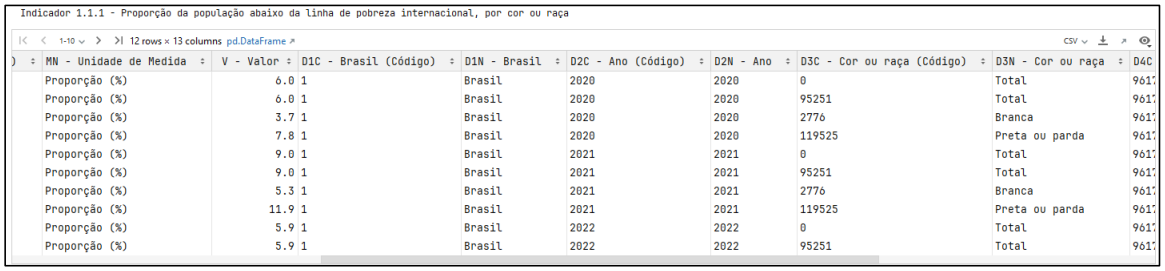
A Figura 13 descreve a estrutura lógica da função “gerar_dataframes_sidra”, ressaltando suas peculiaridades. A fim de melhor exemplificação, a Figura 14 exibe um dos *DataFrames* resultado da utilização da função para o tema “Erradicação da pobreza”.

Figura 13 - Função responsável pela criação do *DataFrame* principal.

```
def gerar_dataframes_sidra(lista_classificacoes):
    lista_sidra_contas = []

    for i in lista_classificacoes:
        dicionario = {}
        lista_nivel = []
        aux_id = i['Id']
        aux_classi = i['Classi_Str']
        aux_nivel = i['Nível_Territorial']
        freq = i['Periodo']

        dicionario['Id'] = aux_id
        dicionario['Nome'] = i['Nome']
        dicionario['Freq'] = freq

        for t in aux_nivel:
            data_sidra = sidrapy.get_table(table_code= f"{aux_id}",
                                territorial_level = f"{t.replace('N','')}",
                                ibge_territorial_code = "all",
                                period = "all",
                                header = "y",
                                format = "pandas",
                                classification=f"{aux_classi}")
            lista_nivel.append({"Nivel": t, "DataFrame": data_sidra})


        dicionario['Data'] = lista_nivel
        lista_sidra_contas.append(dicionario)
```

Fonte: Elaborado pelo autor (2024).

Figura 14 - *DataFrame* presente no item “Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional, por cor ou raça”.



Indicador 1.1.1 - Proporção da população abaixo da linha de pobreza internacional, por cor ou raça

1-10 12 rows × 13 columns pd.DataFrame CSV

)	MN - Unidade de Medida	V - Valor	D1C - Brasil (Código)	D1N - Brasil	D2C - Ano (Código)	D2N - Ano	D3C - Cor ou raça (Código)	D3N - Cor ou raça	D4C
	Proporção (%)	6.0	1	Brasil	2020	2020	0	Total	961
	Proporção (%)	6.0	1	Brasil	2020	2020	95251	Total	961
	Proporção (%)	3.7	1	Brasil	2020	2020	2776	Branca	961
	Proporção (%)	7.8	1	Brasil	2020	2020	119525	Preta ou parda	961
	Proporção (%)	9.0	1	Brasil	2021	2021	0	Total	961
	Proporção (%)	9.0	1	Brasil	2021	2021	95251	Total	961
	Proporção (%)	5.3	1	Brasil	2021	2021	2776	Branca	961
	Proporção (%)	11.9	1	Brasil	2021	2021	119525	Preta ou parda	961
	Proporção (%)	5.9	1	Brasil	2022	2022	0	Total	961
	Proporção (%)	5.9	1	Brasil	2022	2022	95251	Total	961

Fonte: Elaborado pelo autor (2024).

As tabelas criadas não apresentam dados nulos, entretanto é necessário tratar as colunas para que os valores estejam definidos de maneira correta, alterar o nome das mesmas para resultar em uma interface intuitiva e também filtrar para possuírem apenas valores no intervalo de tempo definido, evitando assim gasto de processamento para informações que não serão utilizadas. As Figuras 15 e 16 apresentam esse processo.

Figura 15 - Funções de tratamento das tabelas.

```
def alterar_colunas(lista_data):
    for i in lista_data:
        aux_data = i['Data']
        lista_coluna = []

        for d in aux_data:
            for j in d['DataFrame'].columns:
                coluna = f"{j} - {d['DataFrame'][f'{j}'][0]}"
                lista_coluna.append(coluna)

            d['DataFrame'] = d['DataFrame'] .rename(columns=dict(zip(d['DataFrame'] .columns, lista_coluna)))

def tratar_colunas(lista_data):
    for data in lista_data:
        for data_a in data['Data']:
            data_a['DataFrame'] = data_a['DataFrame'].drop(data_a['DataFrame'].index[0])
            data_a['DataFrame']['D2N - Ano'] = pd.to_datetime(data_a['DataFrame']['D2N - Ano']).dt.year
            data_a['DataFrame']['V - Valor'] = pd.to_numeric(data_a['DataFrame']['V - Valor'], errors='coerce')
```

Fonte: Elaborado pelo autor (2024).

Figura 16 - Filtrando todos *DataFrames* para possuir apenas dados no intervalo destacado.

```
for data in sidra_pobreza:
        aux_freq = data['Freq']
        for d in data['Data']:
            if aux_freq == 'anual':
                d['DataFrame'] = d['DataFrame'].loc[(d['DataFrame']['D2N - Ano'] >= 2020)]

            elif aux_freq == 'mensal':
                d['DataFrame']['V - Valor'] = pd.to_numeric(d['DataFrame']['V - Valor'], errors='coerce')
                d['DataFrame']['D2C - Mês (Código)'] = pd.to_numeric(d['DataFrame']['D2C - Mês (Código)'], errors='coerce')
                d['DataFrame'] = d['DataFrame'].loc[(d['DataFrame']['D2C - Mês (Código)'] >= 202001)]
```

Fonte: Elaborado pelo autor (2024).

Finalizado o armazenamento e tratamento da base de dados verídicos, separados por assuntos, a atenção deve ser tomada para o material que será analisado.

3.3.2 Textos noticiários

Adiante, a atenção deve ser voltada para a seleção dos dados necessários para a concretização da análise desejada. Inicialmente, a ideia principal seria obter textos de redes sociais, principalmente X (o antigo *Twitter*) pela imensa disseminação de falácias que se propaga na plataforma, entretanto com as mudanças das normas das APIs oferecidas pelo antigo *Twitter*, abriu-se portas para buscar textos noticiários diretamente da seção notícias do *Google*.

Para efetuar tal tarefa, foi-se utilizada a API *GoogleNews*. A qual de forma geral, a ferramenta possibilita: buscar notícias a partir de palavras chaves, definir um intervalo de dados da busca dos textos, selecionar a linguagem fonte, apresentar os resultados, dentre outras possibilidades. Para sua utilização deve ser feita sua devida instalação e importação, sendo feito através dos comandos presente na Figura 17.

Figura 17 - Instruções inicias para o uso do *GoogleNews*.

```
[ ]: pip install GoogleNews

[ ]: from GoogleNews import GoogleNews
```

Fonte: Elaborado pelo autor (2024).

Através dos assuntos e intervalo de tempo já definidos anteriormente, se faz possível e necessário agrupar textos noticiários suficientes para progredir com a análise. A Figura 18 demonstra as funções ideais para a utilização da funcionalidade do *GoogleNews*.

Figura 18 - Funções de funcionamento geral do *GoogleNews*.

```
[ ]: googlenews.set_lang(LINGAUGEM)
     googlenews.set_time_range(DATAINICIO,DATAIFIM)
     googlenews.set_encode('utf-8')
     googlenews.search(ASSUNTO)
```

Fonte: Elaborado pelo autor (2024).

Cada chamada utilizando GoogleNews coleta a primeira página do assunto passado como parâmetro de pesquisa. Sendo assim, a função “pesquisa” (Figura 19) é criada para armazenar todas as notícias encontradas pelo período definido e com o limite de 31 páginas em único item. A cada página lida, todas as notícias são armazenadas em um *DataFrame* para posteriormente ocorrer a unificação de todos (excluindo possíveis duplicatas) gerando uma única tabela com todos as notícias encontradas. A função além de retornar a tabela completa, também retorna uma lista de todos os *urls* referentes a cada notícia.

Pelo fato da API realizar inúmeras requisições cada vez que a função é convocada, se faz importante armazenar o *DataFrame* gerado em um arquivo *Excel* (utilizando a própria funcionalidade da biblioteca “pandas”), tal ação impede situações de *flood* (“enchente” de requisições *webs*) para com os servidores do *Google,* caso o *flood* aconteça os servidores “barram” por cerca de um dia as operações disponibilizadas pela API. Sendo assim, a segunda vez que o código será executado o mesmo criará o *DataFrame* a partir de um arquivo salvo na pasta do projeto.

Figura 19 - Função a qual coleta as notícias.

```python
def pesquisas(tema):
    noticias = GoogleNews()
    noticias.setlang('pt')
    noticias.set_time_range('01/01/2020', '01/01/2024')
    noticias.search(tema)
    resultado = noticias.result()
    df = pd.DataFrame(resultado)

    for i in range(2, 31):
        noticias.getpage(i)
        resultado = noticias.result()
        df = df._append(resultado)
        df = pd.DataFrame(df)

    df = df.drop_duplicates(subset=['title'], keep='last')
    df.reset_index(drop=True, inplace=True)
    urls = noticias.get_links()
    return df, urls
```

Fonte: Elaborado pelo autor (2024).

A Figura 20 , estampa a visualização do *DataFrame* formada a partir das notícias relacionadas ao tema de busca “Taxa pobreza IBGE”.

Figura 20 - *DataFrame* notícias para busca “Taxa pobreza IBGE”.

10 rows × 7 columns pd.DataFrame

	title	media	date	datetime	desc	img
0	Pobreza cai em 25 Estados e no D…	O TEMPO	á 19 horas	NaN	Geração de empreg…	… …
1	Segundo IBGE população mais pobr…	A8 Sergipe	á 1 dia	NaN	O Portal A8SE.com…	… …
2	Paraíba registra crescimento aci…	OPovoPB	á 1 dia	NaN	O Instituto Brasi…	… …
3	DESIGUALDADE: IBGE aponta que 1%…	Rondoniaovivo.com	á 3 dias	NaN	Rendimento médio …	… …
4	IBGE: Brasil tem renda recorde, …	A Seguir Niterói	á 3 dias	NaN	Um ano depois do …	… …
5	Pesquisa feita pelo IBGE revela …	R7	á 4 dias	NaN	Em 2023, o 1% mai…	… …
6	Dados do IBGE mostram que abismo…	Sindicato dos Ba…	á 4 dias	NaN	Carlos Vasconcell…	… …
7	Os 10% mais ricos do país têm 14…	Portal AZ	á 4 dias	NaN	Segundo uma ediçã…	… …
8	IBGE: Bolsa Família eleva renda …	Diário do Centro…	á 4 dias	NaN	A expansão do Bol…	… …
9	1% mais rico ganha 39,2 vezes ma…	VG Notícias	á 4 dias	NaN	Em 2023, os 1% da…	… …

Fonte: Elaborado pelo autor (2024).

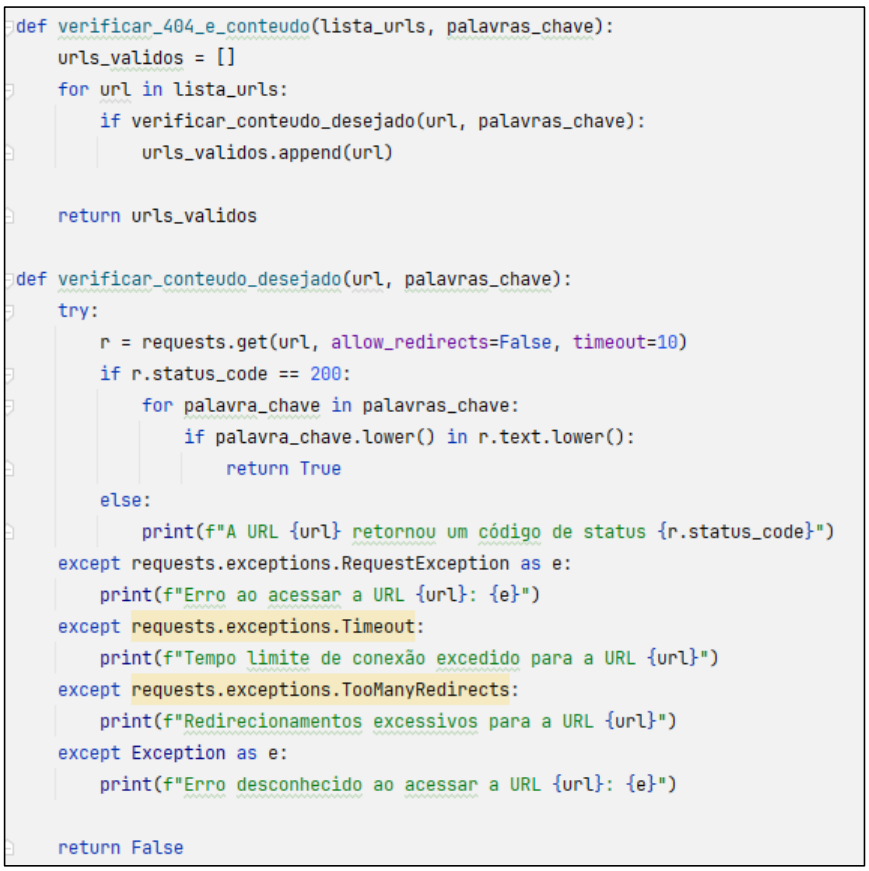
3.4 Tratando textos coletados

O armazenando das notícias possibilita a principal fonte de textos para serem analisados, justamente por esse aspecto é crucial o bom tratamento dos mesmos. A primeira preocupação gira em torno da validação dos *links*: apesar da

API escolhida para retirar textos noticiários, apresentar uma facilidade de uso, a mesma apresenta como ônus a coleta de diversos *links* com erros 404 (página não encontrada) e em outros casos a própria página principal do site, e não a notícia especificada. Fazendo assim necessário limpar *links* que apresentam algum dos dois casos citados.

A função “verificar_404_e_conteudo” trabalha juntamente com uma outra denominada como “verificar_conteudo_desejado”. A lista de *urls* é enviada para a primeira das funções, a mesma lê e envia cada *link* para a segunda função, caso o retorno de “verificar_conteudo_desejado” seja verdadeiro o item se vê sendo adicionado em uma nova lista (a qual será retornada pela função “verificar_404_e_conteudo”).

Figura 21 - Tratando *urls* por funções.



```
def verificar_404_e_conteudo(lista_urls, palavras_chave):
    urls_validos = []
    for url in lista_urls:
        if verificar_conteudo_desejado(url, palavras_chave):
            urls_validos.append(url)

    return urls_validos

def verificar_conteudo_desejado(url, palavras_chave):
    try:
        r = requests.get(url, allow_redirects=False, timeout=10)
        if r.status_code == 200:
            for palavra_chave in palavras_chave:
                if palavra_chave.lower() in r.text.lower():
                    return True
        else:
            print(f"A URL {url} retornou um código de status {r.status_code}")
    except requests.exceptions.RequestException as e:
        print(f"Erro ao acessar a URL {url}: {e}")
    except requests.exceptions.Timeout:
        print(f"Tempo limite de conexão excedido para a URL {url}")
    except requests.exceptions.TooManyRedirects:
        print(f"Redirecionamentos excessivos para a URL {url}")
    except Exception as e:
        print(f"Erro desconhecido ao acessar a URL {url}: {e}")

    return False
```

Fonte: Elaborado pelo autor (2024).

O *url* presente na função “verificar_conteudo_desejado” passa por uma lógica de tratamento de erros. Caso a requisição feita pela biblioteca *requests*,

retornar o status de sucesso (código 200) é verificado se existe a presença de determinada palavra chave no texto (em todos os casos foram utilizados o assunto e o nome do Instituto Brasileiro de Geografia e Estatística). Entretanto, caso a requisição retorne qualquer outro tipo de código, o *url* é apontado como indesejado identificando o possível erro do próprio. A ideia de utilizar a lógica de tratamento de erros, proporciona a identificação de possíveis outros erros como por exemplo de tempo de espera e do excesso de redirecionamentos. A Figura 21 exibe as funções descritas para filtragem dos *urls*.

Notícias em torno da escassez da pobreza, resultaram na presença de 16 *links* válidos, tendo assim mais de 120 *links* inválidos pontuando o ônus anteriormente descrito. Posterior a filtragem, surge a possibilidade e certeza de extrair trechos de cada texto noticiário presente na lista de *urls* já verificados.

Inicialmente a função “pesquisar_texto” é chamada para realizar a extração dos parágrafos do texto noticiário, sendo assim necessário a utilização da biblioteca “*BeautifulSoup*” a qual permite a manipulação, em nível de *Python*, do conteúdo proveniente da Linguagem de Marcação de Hipertexto (HTML) presente em cada *url.* A requisição do *url* especificado é feita para assim utilizar a função referente ao “*BeautifulSoup*”, a qual é definida para retornar todos os elementos de *tag* “<p>” (*tag* essa que define um parágrafo em HTML). O código percorre cada parágrafo para posteriormente unir todos em uma única *string,* retornando uma nova lista, de dicionários, com a nova chave relacionada ao texto completo extraído (MOZILLA DEVELOPER NETWORK, 2024).

A Figura 22 demonstra a montagem da função citada.

Figura 22 - Função que retornar textos de uma página *web* específica.

```
def pesquisar_texto(url):
    lista = []
    v_id = 0
    for i in url:
        textos = []
        dic = {}

        res  = requests.get(i, allow_redirects=False)
        conteudo_html = res.text
        soup = BeautifulSoup(conteudo_html, 'lxml')
        paragrafos = soup.find_all('p')

        for p in paragrafos:
            texto = p.get_text()
            textos.append(texto)

        separador = (f'')
        tex_junto = separador.join(textos)
        v_id = v_id + 1
        dic['Id'] = v_id
        dic['Url'] = i
        dic['Texto'] = tex_junto
        lista.append(dic)
    return lista
```

Fonte: Elaborado pelo autor (2024).

Intrinsicamente com o processo de coleta dos textos disponíveis em cada página, os conceitos em torno do processamento de linguagem natural começam a ter um papel fundamental. A biblioteca escolhida para desempenhar esse papel foi a Spacy principalmente pelo fato da mesma suportar o idioma português. O modelo de linguagem é selecionado a partir da passagem do parâmetro "pt_core_news_md" para a função “spacy.*load”:* o parâmetro se refere ao nome do modelo, que anteriormente necessita ser devidamente instalado. O trecho “md” revela o tamanho do modelo, ou seja, para a análise do projeto foi-se utilizado um modelo intermediário a qual já demonstra uma boa precisão e principalmente, não toma muito tempo de processamento. Sendo assim, para uma primeira implementação foi preferível utilizar um modelo de *pipeline* já treinado priorizando uma boa eficiência.

Além de carregar o modelo de linguagem, a Figura 23 apresenta todas as importações e configurações necessárias para a utilização da biblioteca *Spacy*.

Figura 23 - Configurações iniciais *Spacy*.

```
import spacy
from spacy import displacy
from spacy.matcher import Matcher
from spacy.lang.pt.stop_words import STOP_WORDS
nlp = spacy.load("pt_core_news_md")
stop_words = spacy.lang.pt.stop_words.STOP_WORDS
```

Fonte: Elaborado pelo autor (2024).

A transformação de *string* para uma variável do tipo *doc* (documento) é feito a partir da função “doc_textos”, a qual coleta e processa cada texto presente na lista retornada pela função “processar_texto”. Um elemento do tipo *doc*, apresenta todos os conceitos relacionados a um Processamento de Linguagem Natural convencional: o texto anteriormente em *string* agora possui um texto com tokens (cada palavra ou símbolo presente no texto) e sentenças (conjunto limitado de tokens). A realização da análise e tratamento do texto só se vê possível graças a utilização da função *nlp*, função *essa* nativa do *Spacy*.

A Figura 24 exibe as funções responsáveis pelo processamento dos textos encontrados em cada *url*, já armazenados anteriormente (Figura 22). Além da função realizar o processamento do texto, a mesma também adiciona novas chaves para os itens da lista de parâmetro: ‘Sem_Stop’ e ‘Texto_Filtrado’.

Figura 24 - Função para processamento dos textos.

```
def processar_texto (texto):
    doc = nlp(texto)
    return doc

def doc_textos(lista):
    for i in lista:
        sep = processar_texto(i['Texto'])
        sem_stop = remove_stop_words(sep)
        i['Texto'] = sep
        i['Sem_Stop'] = sem_stop
        i['Texto_Filtrado'] = filtrar_palavras(sep)
```

Fonte: Elaborado pelo autor (2024).

A primeira chave irá receber o texto processado sem a presença de *stopwords*, através da função “remove_stop_words*”*. *Stopwords* em suma, se caracterizam como palavras irrelevantes que não alteram no entendimento de uma palavra ou frase (ANTIĆ, 2021). O funcionamento dessa etapa em nível de código, é realizado através da verificação de cada token, armazenando aqueles que não são uma *stopword*, para logo após retornar a junção dos tokens já devidamente processados (estrutura do processo presente na Figura 25).

O valor referente a chave ‘Texto_Filtrado’ será descrito no capítulo de conclusão, já que se revela como uma tentativa de filtragem descartada.

Figura 25 - Funções para remover *stopwords* e a tentativa de filtragem.

```
def remove_stop_words(sentenca):
  doc = nlp(sentenca)
  tokens_filtrados = [token for token in doc if not token.is_stop]
  return nlp(' '.join([token.text for token in tokens_filtrados]))

def filtrar_palavras(setenca):
    doc = nlp(setenca)
    # Lista de categorias a serem mantidas
    categorias_mantidas = ["NOUN", "PROPN", "ADJ", "SYM", "PUNCT", "NUM"]
    tokens_filtrados = [token.text for token in doc if token.pos_ in categorias_mantidas]
    return nlp(" ".join(tokens_filtrados))
```

Fonte: Elaborado pelo autor (2024).

O resultado esperado das funções de processamento de linguagem natural, possibilita a primeira grande filtragem que o material precisa passar. A função “trechos_chaves” visa retornar os possíveis trechos que citam especificamente um dado ou o próprio IBGE.

A Figura 26 destaca a função responsável pela filtragem, função essa que será descrita posteriormente.

Cada item da lista é guardado, recolhendo assim seu texto relacionado para ser a base da busca de trechos chaves. A biblioteca *Spacy* possibilita a criação de regras a qual definem a estrutura que um texto deve seguir, regras essas que são por muitas vezes especificadas pelos atributos dos tokens (SPACY, 2024). A primeira regra definida se relaciona para que obrigatoriamente uma sentença do texto deva possuir um token que apresente um tipo “LOC” (localização) de valor IBGE (utiliza-se LOWER para realizar a comparação em uma expressão

inteiramente composta por letras minúsculas). Sendo assim primeiramente é verificado a presença da citação do Instituto Brasileiro de Geografia e Estatística. A segunda regra, especifica um texto que apresente algum valor numérico seguido do símbolo de “%”.

Figura 26 - Funções para filtragem de trechos chaves de cada texto.

```
def trechos_chaves(lista):
    retorno = []
    matcher = Matcher(nlp.vocab)
    padrao_ibge = [
        {"ENT_TYPE": "LOC", "LOWER": "ibge"}
    ]
    porcentagem = [
        {"TEXT": {"REGEX": r"\d+(?:[\.,]\d+)?%?"}}
    ]

    # Adicionar padrão ao Matcher
    matcher.add("IBGE", [padrao_ibge])
    # Adicionar padrão ao Matcher
    matcher.add("Porcentagem", [porcentagem])

    for i in lista:
        dic = {'Id': i['Id'], 'Url': i['Url'], 'Texto': set(), 'Sem_Stop': set()}
        match = matcher(i['Texto'])
        match1 = matcher(i['Sem_Stop'])
        match2 = matcher(i['Texto_Filtrado'])
        sents = set()
        sents_stop = set()
        sents_filtrado = set()

        for match_id, start, end in match:
            sentence = i['Texto'][start:end].sent
            sents.add(sentence)

        for match_id, start, end in match1:
            sentence1 = i['Sem_Stop'][start:end].sent
            sents_stop.add(sentence1)

        for match_id, start, end in match2:
            sentence1 = i['Texto_Filtrado'][start:end].sent
            sents_filtrado.add(sentence1)

        dic['Texto'] = sents
        dic['Sem_Stop'] = sents_stop
        dic['Texto_Filtrado'] = sents_filtrado
        retorno.append(dic)
    return retorno
```

Fonte: Elaborado pelo autor (2024).

O *matcher* (função atribuída ao “*spacy”*) adiciona ambas regras, criando uma relação de que será buscado em cada texto, trechos que correspondam a pelo menos alguma das regras criadas. Cada trecho encontrado é salvo em um

conjunto (*set)* para evitar possíveis valores repetidos. A filtragem é feita para cada texto processado, presentes nas chaves específicas de cada item.

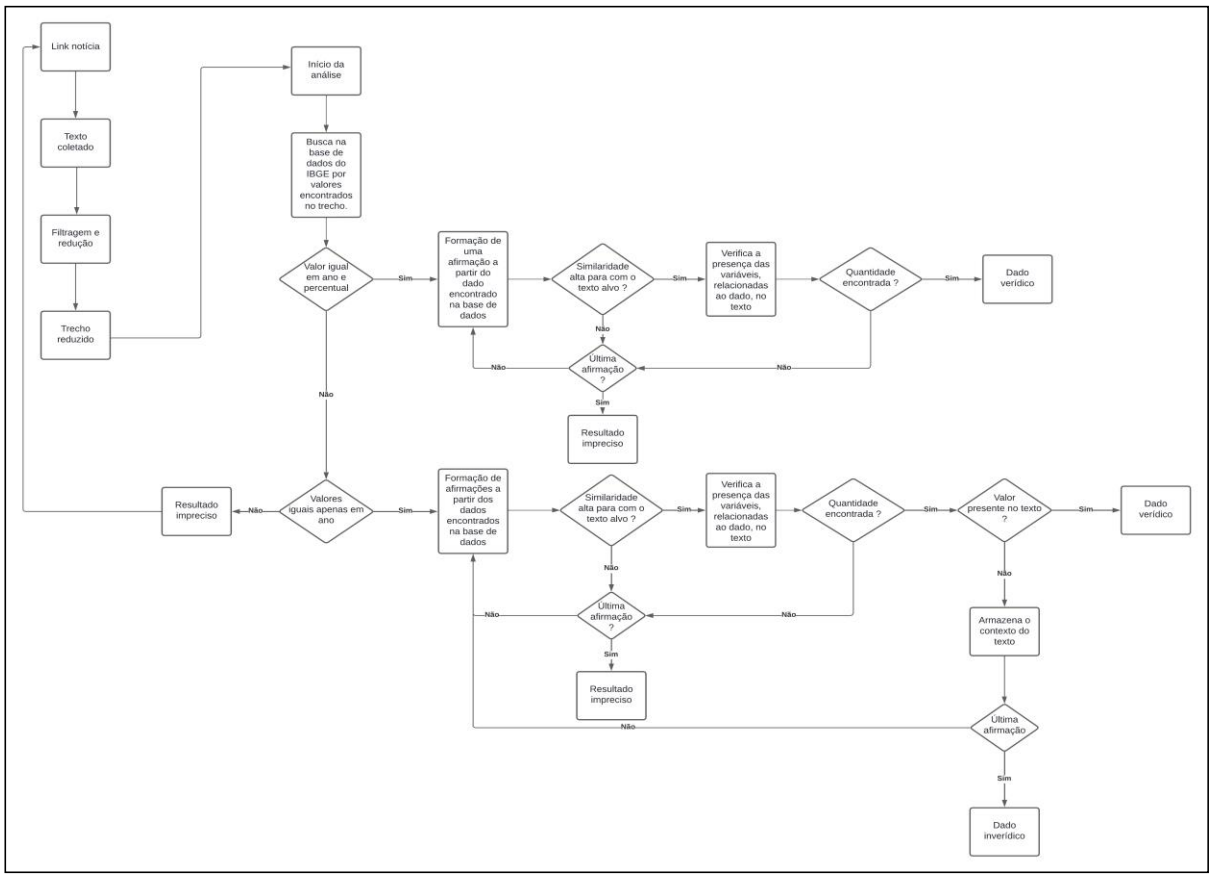
Por fim, a função irá retornar uma nova lista (feita apenas para melhor visualização de progressão) com os valores conhecidos, porém, agora filtrados.

3.5 Análise dos textos

Analisar a relação de um determinado dado presente em um texto com os resultados já estabelecidos de pesquisas, demonstra ser um processo minucioso. O fluxograma exibido na Figura 27, reflete as etapas, de forma geral, para a análise de cada texto extraído. Cada etapa será descrita detalhadamente ao decorrer do tópico presente.

Embora o texto extraído já tenha passado por uma filtragem, se faz necessário diminuir ainda mais o escopo ao mesmo tempo confirmar o contexto dos mesmos, concentrando assim em selecionar todo trecho que possuí uma certa similaridade com os temas dos *DataFrames* das pesquisas armazenadas do IBGE.

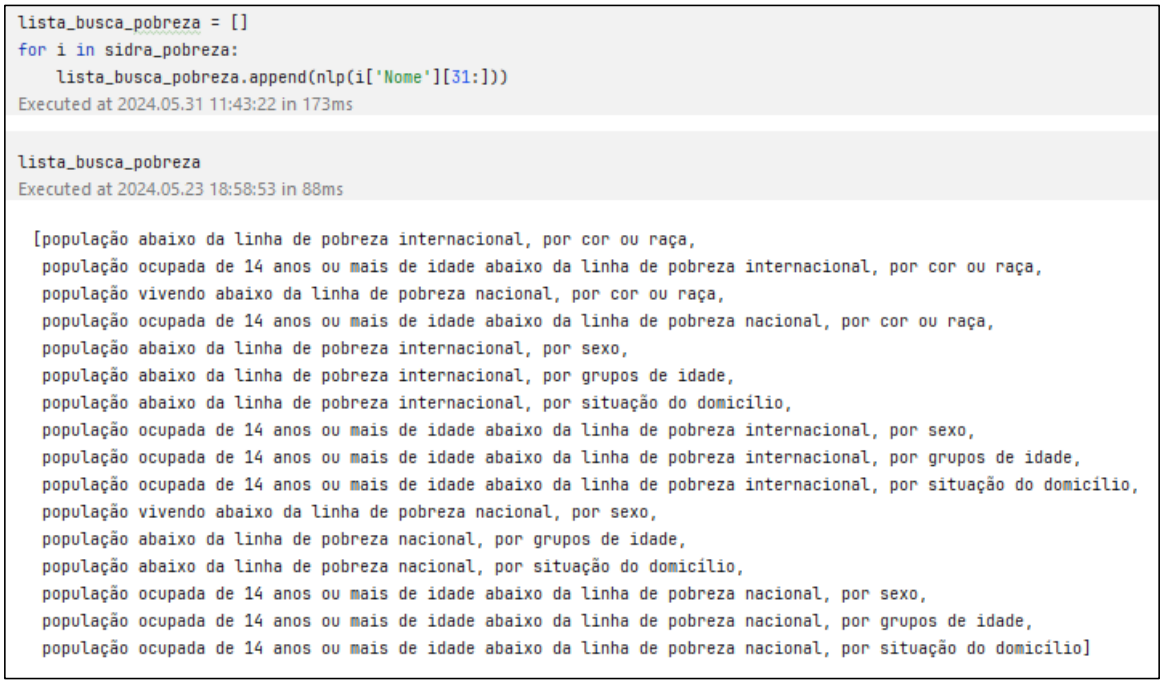
Figura 27 - Fluxograma simplificado da análise de um texto.



Fonte: Elaborado pelo autor (2024).

A Figura 28 evidencia o primeiro passo para a redução do escopo para a análise. Nessa etapa, o conjunto de resultados é percorrido armazenando cada tema, descrito em um item, em uma nova lista. A *string* correspondente do tema é processada para melhor utilização posteriormente. Subsequentemente a lista é posta como parâmetro para a função “filtro_assunto”.

Figura 28 - Lista com o nome das pesquisas do assunto “Erradicação da pobreza”.



```
lista_busca_pobreza = []
for i in sidra_pobreza:
    lista_busca_pobreza.append(nlp(i['Nome'][31:]))
Executed at 2024.05.31 11:43:22 in 173ms

lista_busca_pobreza
Executed at 2024.05.23 18:58:53 in 88ms

[população abaixo da linha de pobreza internacional, por cor ou raça,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional, por cor ou raça,
 população vivendo abaixo da linha de pobreza nacional, por cor ou raça,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza nacional, por cor ou raça,
 população abaixo da linha de pobreza internacional, por sexo,
 população abaixo da linha de pobreza internacional, por grupos de idade,
 população abaixo da linha de pobreza internacional, por situação do domicílio,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional, por sexo,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional, por grupos de idade,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional, por situação do domicílio,
 população vivendo abaixo da linha de pobreza nacional, por sexo,
 população abaixo da linha de pobreza nacional, por grupos de idade,
 população abaixo da linha de pobreza nacional, por situação do domicílio,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza nacional, por sexo,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza nacional, por grupos de idade,
 população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza nacional, por situação do domicílio]
```

Fonte: Elaborado pelo autor (2024).

O código proveniente da Figura 29, percorre todos os textos armazenados, comparando cada sentença com cada tema das pesquisas. Para concretizar a comparação, a função *'similarity'* é utilizada, sendo passado como parâmetro uma sentença do texto completo e cada assunto de pesquisa. Proveniente da biblioteca *Spacy, 'similarity'* analisa a sintaxe e semântica de cada parâmetro, encontrando um valor que simbolize o quão similar ambas partes são entre si. Valores próximos de 1 acentuam uma forte similaridade, enquanto valores próximos de 0 apontam a fraca relação entre os textos. A eficácia da metodologia está relacionada diretamente com o modelo de *pipeline* escolhido, a precisão da comparação se relaciona pontualmente com o tamanho do modelo. Mesmo o projeto não fazendo uso do modelo de maior escala, para uma filtragem simples de contexto traz resultados coerentes. Para a sentença ser escolhida, o resultado da similaridade deve ser minimante igual a 0.65. O valor definido é resultado de testes: por se tratar de uma sentença maior, consequentemente, com mais conteúdo comparado a apenas uma frase que define um tema, não se torna comum valores elevados (igual ou acima de 0.7). Logo, buscando a não ocorrência de valores falsos

negativos, o valor condicional se torna inferior. Para o caso da erradicação da pobreza, 3 textos foram descartados por completo.

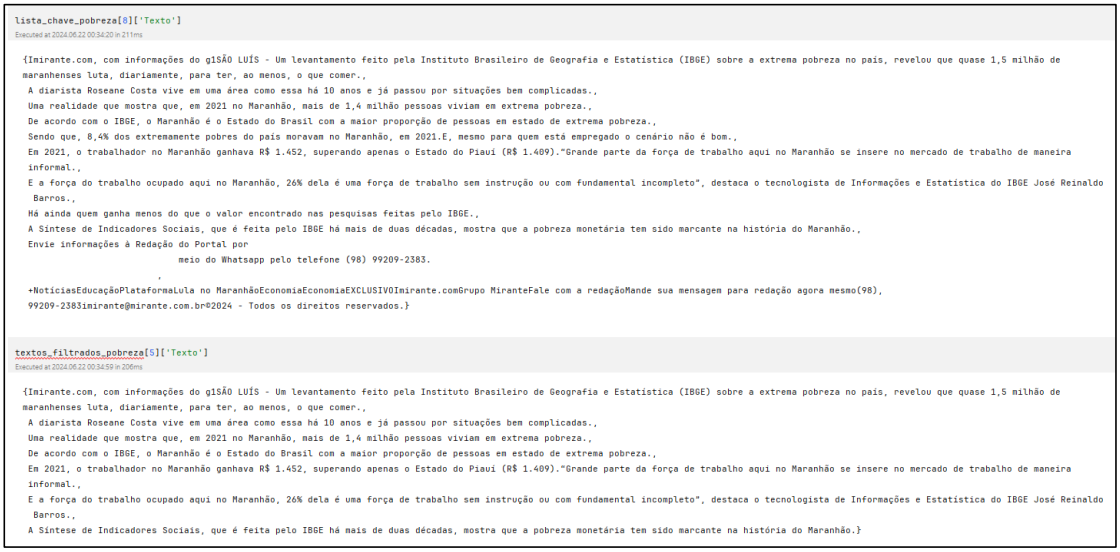
Figura 29 - Função para filtragem de trechos relacionados as pesquisas.

```
def filtro_assunto(lista_itens, lista_assunto):
    retorno = []
    for valor in lista_itens:
        valor_aux = valor
        sentencas_unicas = set()
        texto = valor['Texto']

        for sents in texto:
            for comp in lista_assunto:
                similaridade = comp.similarity(sents.sent)
                if similaridade > 0.65:
                    sentencas_unicas.add(sents.sent)
                if comp.text in sents.sent.text:
                    sentencas_unicas.add(sents.sent)

        if sentencas_unicas:
            string_aux = ' '.join([span.text for span in sentencas_unicas])
            sem_stop = remove_stop_words(string_aux)
            texto_filtrado = filtrar_palavras(string_aux)

            valor_aux['Texto'] = sentencas_unicas
            valor_aux['Sem_Stop'] = sem_stop
            valor_aux['Texto_Filtrado'] = texto_filtrado

            retorno.append(valor_aux)

    return retorno
```

Fonte: Elaborado pelo autor (2024).

A Figura 30 demonstra um exemplo prático da filtragem mostrando um texto antes e após a aplicação da função “filtro_assunto”.

Figura 30 - Exemplo aplicação função filtro_assunto.



Fonte: Elaborado pelo autor (2024).

A certeza do contexto de cada trecho destacado, possibilita a extração dos valores alvos: um dado percentual e também o ano referente. O resultado demonstra a forma mais reduzida do texto, mantendo apenas o necessário para realizar a consulta nos *DataFrames* da base de dados do IBGE. Visar a efetividade da etapa destacada, se faz necessário dividi-la em 2 partes:

- Primeiramente ocorre a chamada da função “trechos_busca”: o texto presente em cada item da lista enviada como parâmetro, se vê analisado com intuito de extrair um dado com estrutura de um ano e de um valor percentual. Dessa maneira, cada token de cada sentença é percorrido verificando a sua devida estrutura. Caso seja um número de tamanho semelhante a um ano, o dado é devidamente armazenado, também é conferido se o token apresenta um “.” em sua sequência, isso ocorre para o algoritmo ser capaz de coletar tanto um dado semelhante a “2020” quanto um outro próximo a “2020.”. Além de se atentar para o tamanho do valor numérico, a função também verifica e coleta uma data com o formato de mês e ano, assim como uma data no formato “12/02/2022”. Por último, a cada iteração, também é armazenado qualquer valor numérico, qualquer valor referente a “%” e qualquer dado do tipo “LOC” (tipo que corresponde a uma localização, a

descrição é feita ao realizar o Processamento de Linguagem Natural). Com todos os identificados, cada dicionário da lista recebe uma nova chave “Selecionados”, como valor é definido os dados específicos selecionados e sua respectiva sentença. A função também retorna uma nova lista a qual recebe a cópia do resultado obtido.

- Os dados antes selecionados, são tratados e salvos em uma lista de dicionários com as seguintes chaves: “Dado” (valor numérico juntamente com a sua %), “Ano” (apenas anos de 2020 para frente), “Id” (com o respectivo Id do texto fonte) e se possuir uma localização também é criada a chave “Loc”. A função “dados_para_comparacao” retornará apenas os valores em uma maneira de fácil distinção e utilização.

As funções são descritas na Figura 31 e 32 respectivamente. Já a Figura 33, exibe um exemplo prático do retorno resultante do conjunto de ambas funções.

Figura 31 - Função responsável em selecionar dados específicos.

```
def trechos_busca(lista_val):
    trechos_selecionados = []
    anos = ['2018', '2019', '2020', '2021', '2022', '2023', '2024', '2018.', '2019.', '2020.', '2021.', '2022.', '2023.', '2024.']
    for item in lista_val:
        item_aux = item
        texto_aux = item_aux['Texto']
        trechos_por_sentenca = []

        for sents in texto_aux:
            trechos = []

            for token in sents:
                if token.like_num and token.text.isnumeric() or token in anos:
                    if len(token.text) == 4 or (len(token.text) == 5 and token.text in anos):  # Ano
                        if token.i > 0:
                            trechos.append(token.nbor(-1).text + " " + token.text)
                elif token.text.lower() in ["janeiro", "fevereiro", "março", "abril", "maio", "junho", "julho", "agosto","setembro", "outubro", "novembro", "dezembro"]:
                    if token.i > 0 and token.i < len(sents):
                        if sents[token.i - 1].like_num and sents[token.i - 1].text.isnumeric():  # Mês e ano
                            trechos.append(sents[token.i - 1].text + " " + token.text)
                elif token.text == "/":
                    if token.i > 0 and token.i < len(sents) - 1:
                        if sents[token.i - 1].like_num and sents[token.i + 1].like_num:
                            trechos.append(sents[token.i - 1].text + "/" + sents[token.i + 1].text + "/" + sents[token.i + 2].text)

            for token in sents:
                if token.pos_ == "SYM" and "%" in token.text:
                    trechos.append(token.text)
                if token.like_num:
                    trechos.append(token.text)
                if token.ent_type_ == "LOC":
                    trechos.append(token.text)

            trechos_por_sentenca.append([sents.text, trechos])

        item_aux['Selecionados'] = trechos_por_sentenca
        trechos_selecionados.append(item_aux)
    return trechos_selecionados
```

Fonte: Elaborado pelo autor (2024).

Figura 32 - Função dados_para_comparacao.

```python
def dados_para_comparacao(trechos):
    list_comp = []
    anos = ['2018', '2019', '2020', '2021', '2022', '2023', '2024', '2018.', '2019.', '2020.', '2021.', '2022.', '2023.', '2024.']
    for item in trechos:
        selecionados = item['Selecionados']
        for parte in selecionados:
            busca = {}
            trechos = parte[1]
            if '%' or 'cento' or '%.' in trechos:
                for token in trechos:
                    try:
                        if int(token) >= 2020:
                            ano = int(token)
                            busca['Ano'] = ano
                    except ValueError:
                         if token in anos and '.' in k:
                            ano = token.replace('.', '')
                            ano = int(ano)
                            if int(token) >= 2020:
                                busca['Ano'] = ano
                            pass

                    if nlp(token)[0].ent_type_ == "LOC":
                        loc = token
                        busca['Loc'] = loc

                    if token == '%' or (token == '.' and trechos[(trechos.index(token)) - 1] == '%'):
                        indice = (trechos.index(token))
                        valor = trechos[indice-1] + trechos[indice]
                        busca['Dado'] = valor

            if 'Dado' in busca:
                busca['Id'] = item['Id']
                list_comp.append(busca)
    return list_comp
```

Fonte: Elaborado pelo autor (2024).

Figura 33 - Trecho do retorno da chamada em conjunto das funções.

comp_pobreza

Executed at 2024.05.23 19:00:49 in 288ms

```
[{'Loc': 'IBGE', 'Dado': '1%', 'Ano': 2022, 'Id': 1},
 {'Dado': '63%', 'Loc': 'Brasil', 'Id': 1},
 {'Loc': 'Brasil', 'Dado': '11,5%', 'Ano': 2023, 'Id': 1},
 {'Ano': 2020, 'Dado': '46%', 'Loc': 'Covid-19', 'Id': 3},
 {'Dado': '38,7%', 'Ano': 2022, 'Id': 3},
 {'Loc': 'Brasil', 'Ano': 2022, 'Dado': '54,1%', 'Id': 3},
 {'Dado': '36,7%', 'Ano': 2021, 'Loc': 'Brasil', 'Id': 3},
 {'Ano': 2021, 'Dado': '54,1%', 'Id': 3},
 {'Loc': 'Brasil', 'Dado': '36,7%', 'Ano': 2022, 'Id': 4},
 {'Ano': 2021, 'Dado': '2,4%', 'Id': 4},
 {'Loc': 'Brasil', 'Dado': '1,8%', 'Id': 4},
 {'Dado': '27,4%', 'Id': 4},
 {'Dado': '67%', 'Ano': 2022, 'Id': 4},
 {'Dado': '80%', 'Ano': 2022, 'Id': 4},
 {'Dado': '100%', 'Loc': 'Cras', 'Id': 4},
 {'Loc': 'Brasil', 'Dado': '1,8%', 'Id': 4},
 {'Loc': 'Cruz', 'Dado': '50%', 'Id': 6},
 {'Dado': '58,7%', 'Id': 6},
 {'Dado': '5%', 'Id': 6},
 {'Ano': 2022, 'Loc': 'País', 'Dado': '5%', 'Id': 7},
```

Fonte: Elaborado pelo autor (2024).

Com base nos resultados obtidos pelas funções descritas anteriormente, torna-se possível iniciar a análise comparativa dos dados.

Prontamente cada sentença retornada é analisada, sendo validada apenas as que possuem um conjunto de dado percentual e um ano. Sequencialmente, o atributo “loc” da biblioteca pandas é utilizado para tentar retornar um resultado dos *DataFrames* de todas as pesquisas do assunto especificado. O resultado deverá possuir o mesmo ano e o mesmo valor definido pelo conjunto de dados. Caso o resultado não seja vazio, o próprio será implementado em uma lista com valores em formato de dicionários.

A existência de ao menos um valor não nulo na lista, modifica pontualmente os próximos passos, resultando em uma análise dividida: uma apenas para os itens com respostas semelhantes no *DataFrame* da base, e outra focada em analisar os itens restantes.

3.5.1 Respostas semelhantes encontradas

Inicialmente cada item presente na lista de resultados semelhantes são lidos, armazenando seu respectivo Id. A partir do identificador, é verificado qual elemento da lista de textos noticiários (anteriormente tratados) corresponde com o respectivo valor armazenado.

Resultando em um valor não nulo, o texto relacionado ao Id se vê armazenado em uma variável auxiliar, assim como as respostas equivalentes encontradas nos *DataFrames* e também o valor destaque (o alvo da análise). Todas as respostas equivalentes encontradas nos *DataFrames* são percorridas e devidamente alteradas para uma representação em formato de *string* ao invés de *DataFrame*, dessa maneira o resultado que anteriormente estava disposto em uma tabela é armazenado separadamente em diferentes variáveis. Em posse das variáveis, as mesmas são remanejadas com intuito de gerar uma frase com a seguinte composição: “{Variável apresentada na tabela} é dado(a) por {Valor percentual encontrado}%, a partir do(a) {Tipo do dado } {Valor do tipo} no {Local} e no ano de {Ano}”.

Figura 34 - Função geradora de uma frase com variáveis da resposta.

```
def formar_frase_completa_(lista):
    val_relacionado = []
    juntar = 0
    for i in lista:
        dic = {}
        if 'D1N' in i['Coluna']:
            juntar += 1
            localizacao = i['Valor']
        if ('V - ' or 'V-') in i['Coluna']:
            juntar += 1
            valor_percentual = i['Valor']
        if 'D2N' in i['Coluna']:
            juntar += 1
            ano_data = i['Valor']
        if 'D3N' in i['Coluna']:
            juntar += 1
            tipovar1_data = str(i['Coluna']).replace("D3N -", "").strip().lower()
            tipovar2_data = i['Valor'].lower()
        if 'D4N' in i['Coluna']:
            juntar += 1
            variavel_data = i['Valor']
        if juntar == 5:
            juntar = 0
            if 'Brasil' in localizacao:
                dic['Frase'] = f'{variavel_data} é dado(a) por {valor_percentual}%, a partir do(a) {tipovar1_data} {tipovar2_data} no {localizacao} e no ano de {ano_data}'
                dic['Valor'] = str(valor_percentual).replace('.', ',')
                dic['Variavel'] = variavel_data
                dic['Tipo'] = tipovar1_data
                dic['TipoV'] = tipovar2_data
                dic['Ano'] = ano_data
                dic['Local'] = localizacao
                val_relacionado.append(dic)
            else:
                dic['Frase'] = f'{variavel_data} é dado por {valor_percentual}%, a partir do(a) {tipovar1_data} {tipovar2_data} no {localizacao} e no ano de {ano_data}'
                dic['Valor'] = str(valor_percentual).replace('.', ',')
                dic['Variavel'] = variavel_data
                dic['Tipo'] = tipovar1_data
                dic['TipoV'] = tipovar2_data
                dic['Ano'] = ano_data
                dic['Local'] = localizacao
                val_relacionado.append(dic)
    return val_relacionado
```

Fonte: Elaborado pelo autor (2024).

A Figura 34 apresenta a função geradora da frase especificada anteriormente.

Segue um exemplo de saída da função “formar_frase_completa_” para o caso de apenas um valor presente no *DataFrame* como entrada: [{'Frase': 'Proporção da população abaixo da linha de pobreza internacional é dado por 5.0%, a partir do(a) grupo de idade 25 a 29 anos no Brasil e no ano de 2022', 'Valor': '5,0', 'Variavel': 'Proporção da população abaixo da linha de pobreza internacional', 'Tipo': 'grupo de idade', 'TipoV': '25 a 29 anos', 'Ano': 2022, 'Local': 'Brasil'}].

Através da frase montada, novamente se faz útil a utilização da comparação a partir da similaridade entre dois trechos. No caso específico, se faz evidenciado a similaridade entre o texto do trecho a qual possui o dado analisado e a frase montada. A análise para classificação se o dado está distorcido ou não, exige uma similaridade superior a 0.85. Com intuito de aumentar a assertividade, não se faz suficiente depender apenas do fator similaridade. Consequentemente,

ocorre a verificação da presença do tipo do grupo da pesquisa e de seu respectivo valor (ambos dados provenientes da frase formulada anteriormente) na sentença analisada. Cada valor encontrado é salvo em uma nova lista a qual desempenha um papel de *flag*: se houver mais de 2 valores encontrados e se o valor presente no texto for o mesmo comparado a resposta encontrado do *DataFrame*, este dado é classificado como verídico salvando todas as informações relacionadas em um dicionário.

Entretanto, caso o dado presente no texto não encontrar um resultado verídico, o código classifica a análise como sendo imprecisa. Tal resultado ocorre por dois motivos: o contexto do dado no texto não se mostra pontualmente semelhante a base de dados salva ou apresenta um valor não presente na base de dados. Não sendo possível confirmar com exatidão a análise.

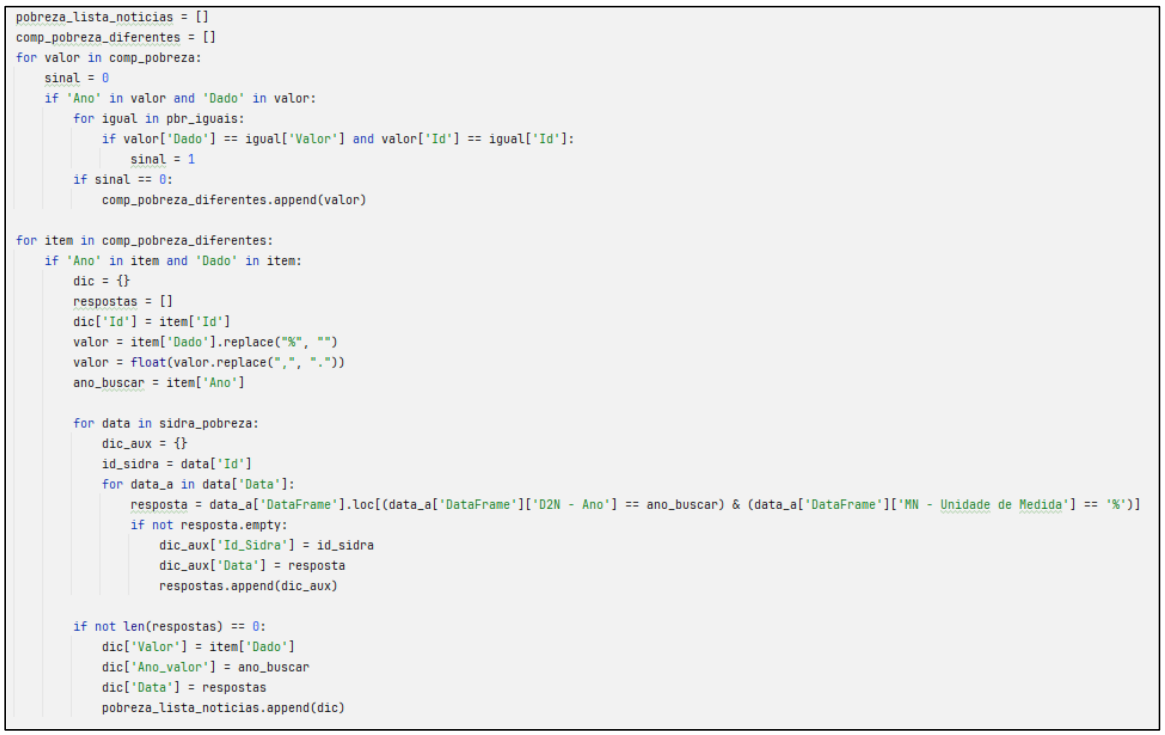
A Figura 35 destrincha o código utilizado para analisar dados que anteriormente encontraram valores similares na base de dados do IBGE, para o assunto “Erradicação da pobreza” (o código não sofre mudanças significativas considerando o mesmo contexto com um outro assunto).

Figura 35 - Análise para valores semelhantes em ano e valor, encontrados nos DataFrames para o assunto “Erradicação da pobreza”.

```
for igual in pbr_iguais:
    id = igual['Id']
    index_trecho_pbr = buscar_trecho(id, textos_filtrados_pobreza)

    if index_trecho_pbr is not None:
        texto_comparar_pbr = textos_filtrados_pobreza[index_trecho_pbr]['Texto']
        item_pbr = textos_filtrados_pobreza[index_trecho_pbr]
        texto_completo_pbr = nlp(sentencas_unidas(texto_comparar_pbr))

        respostas_data_pbr = igual['Data']
        valor_trecho_pbr =  str(igual['Valor']).replace('.', ',')
        flag_pbr = 0
        resposta_flag_pbr = 0

        for sent in texto_comparar_pbr:
            if valor_trecho_pbr in sent.text:
                flag_pbr = 1
                sent_encontrada = sent

        for data in respostas_data_pbr:
            id_sidra = data['Id_Sidra']
            sidra_original = sidra_pobreza[buscar_trecho(id_sidra, sidra_pobreza)]
            lista_frase_pbr, val_frase_pbr = valores_dfs_string(data['Data'])
            frases_pbr = formar_frase_completa_(lista_frase_pbr)


            for valor in frases_pbr:
                similaridade_pbr = texto_completo_pbr.similarity(nlp(valor['Frase']))
                lista_palavras_pbr = []

                if similaridade_pbr > 0.85:
                    dic_resul = {}
                    for token_frase in remove_stop_words(valor['Tipo']):
                        if token_frase.text in sent_encontrada.text:
                            lista_palavras_pbr.append(token_frase)

                    for token_colun in remove_stop_words(valor['TipoV']):
                        if token_colun.text in sent_encontrada.text:
                            lista_palavras_pbr.append(token_colun)

                    if (len(lista_palavras_pbr) >= 2) and flag_pbr == 1:
                        resposta_flag_pbr = 1
                        id_novo += 1
                        url = item_pbr['Url']
                        site_nome = urlparse(url).netloc
                        dic_resul['Id'] = id_novo
                        dic_resul['Site'] = site_nome
                        dic_resul['Url'] = url
                        dic_resul['Texto alvo'] = texto_completo_pbr.text
                        dic_resul['Trecho destaque'] = sent_encontrada.text
                        dic_resul['Valor destacado'] = igual['Valor']
                        dic_resul['Trecho encontrado'] = valor['Frase']
                        dic_resul['DataFrame respectivo'] = id_sidra
                        dic_resul['Resultado'] = 'Dado verídico'
                        resultados_pobreza.append(dic_resul)
                        break

        if flag_pbr != 0 and resposta_flag_pbr == 0:
            id_novo += 1
            url = item_pbr['Url']
            site_nome = urlparse(url).netloc
            dic_resul['Id'] = id_novo
            dic_resul['Site'] = site_nome
            dic_resul['Url'] = url
            dic_resul['Texto alvo'] = texto_completo_pbr.text
            dic_resul['Trecho destaque'] = sent_encontrada.text
            dic_resul['Valor destacado'] = igual['Valor']
            dic_resul['Trecho encontrado'] = None
            dic_resul['DataFrame respectivo'] = None
            dic_resul['Resultado'] = 'Imprecisão na validação, base de dados não suporta'
            resultados_pobreza.append(dic_resul)

        elif flag_pbr == 0:
            print('Dado não está presente no texto')
```

Fonte: Elaborado pelo autor (2024).

Figura 36 - Organização dos dados semelhantes quanto ao ano.



```
pobreza_lista_noticias = []
comp_pobreza_diferentes = []
for valor in comp_pobreza:
    sinal = 0
    if 'Ano' in valor and 'Dado' in valor:
        for igual in pbr_iguais:
            if valor['Dado'] == igual['Valor'] and valor['Id'] == igual['Id']:
                sinal = 1
        if sinal == 0:
            comp_pobreza_diferentes.append(valor)

for item in comp_pobreza_diferentes:
    if 'Ano' in item and 'Dado' in item:
        dic = {}
        respostas = []
        dic['Id'] = item['Id']
        valor = item['Dado'].replace("%", "")
        valor = float(valor.replace(",", "."))
        ano_buscar = item['Ano']

        for data in sidra_pobreza:
            dic_aux = {}
            id_sidra = data['Id']
            for data_a in data['Data']:
                resposta = data_a['DataFrame'].loc[(data_a['DataFrame']['D2N - Ano'] == ano_buscar) & (data_a['DataFrame']['MN - Unidade de Medida'] == '%')]
                if not resposta.empty:
                    dic_aux['Id_Sidra'] = id_sidra
                    dic_aux['Data'] = resposta
                    respostas.append(dic_aux)

        if not len(respostas) == 0:
            dic['Valor'] = item['Dado']
            dic['Ano_valor'] = ano_buscar
            dic['Data'] = respostas
            pobreza_lista_noticias.append(dic)
```

Fonte: Elaborado pelo autor (2024).

Posteriormente, se faz necessário analisar todos os dados que possuem uma resposta semelhante apenas quanto ao ano, comparando com a base de dados do IBGE. A nova seleção exclui os dados já analisados no passo anterior. Estrutura presente na Figura 36.

O código para essa etapa se mostra semelhante ao anterior, possuindo mudanças pontuais e a inserção de uma nova lógica para coletar informações falsas.

No atual cenário, a primeira mudança parte da inserção de um novo *flag* responsável em apontar a ocorrência de uma informação falsa. O mesmo se torna útil para evitar uma interrupção sem antes verificar todos os cenários possíveis a concordância da análise dos dados, ou seja, para julgar uma informação como uma inverdade, antes se faz necessário realizar uma busca completa mesmo que no início ocorra suspeitas de se tratar de uma informação falsa.

Cada dado presente no *DataFrame* de resposta também passa pelo mesmo processo de criação de frases em formato de *string*. A diferença está no fato da lógica ser alterada para a similaridade ocorrer a partir da parte do texto a qual o dado analisado se encontra, não sendo utilizado uma similaridade para com

o texto completo. A alteração parte da ideia de ganhar mais velocidade e aumentar ainda mais os valores obtidos, podendo assim surgir uma margem de 0.02 a mais do que a anterior utilizada. O requisito sendo obedecido, a validação do dado também é feita a partir da presença do contexto proveniente do tipo de grupo da pesquisa e de seu valor.

Observar o assunto “Erradicação da pobreza” também revela uma mudança quanto ao acréscimo de mais um método para filtragem: todos os indicadores de assuntos trazem uma percepção da pesquisa em âmbito nacional ou internacional. Sabendo do padrão presente na pesquisa, durante a análise é verificado se há presença de algum termo similar a “nacional” ou “internacional” no texto levantado para análise. O processo é feito por uma simples busca em uma lista de termos, a utilidade se mostra ao evitar percorrer dados que poderiam ser incoerentes, ganhando mais velocidade na análise. Entretanto, caso nenhuma condição seja cumprida, a análise ocorre de forma completa.

Por fim, se um dado não possui um contexto semelhante para analisar sua veridicidade, o mesmo é apontado como incapaz de possuir uma boa precisão de análise. A Figura 37 representa o trecho responsável em continuar a análise.

Figura 37 - Análise para os demais textos do tema “Erradicação da pobreza”.

```
exp_nacionais = ['taxa brasileira', 'brasileira', 'nacional', 'taxa nacional', 'taxa nacionais','Taxa brasileira', 'Taxa nacional', 'Taxas brasileiras',
exp_internacionais = ['taxa internacional', 'internacional', 'taxas internacionais', 'Taxa internacional', 'Taxas internacionais']
for item in pobreza_lista_noticias:
    id_fake = item['Id']
    index_trecho_fake = buscar_trecho(id_fake, textos_filtrados_pobreza)

    if index_trecho_fake is not None:
        texto_comparar_fake = list(textos_filtrados_pobreza[index_trecho_fake]['Texto'])
        item_fake = textos_filtrados_pobreza[index_trecho_fake]
        texto_completo_fake = nlp(sentencas_unidas(texto_comparar_fake))
        respostas_data_fake = item['Data']
        valor_trecho_fake =  str(item['Valor']).replace('.', ',')
        sent_encontrada_fake = None
        flag_pbr = 0
        resposta_flag_pbr = 0
        resposta_flag_fake = 0
        for sent in texto_comparar_fake:
            if valor_trecho_fake in sent.text:
                sent_encontrada_fake = sent
                flag_pbr = 1
                break

        if sent_encontrada_fake is not None:
            for data in respostas_data_fake:
                if resposta_flag_pbr == 1:
                    break

                id_sidra = data['Id_Sidra']
                lista_frase_fake, val_frase_fake = valores_dfs_string(data['Data'])
                frase_fake = formar_frase_completa_(lista_frase_fake)

                for valor in frase_fake:
                    similaridade_fake = sent_encontrada_fake.similarity(nlp(valor['Frase']))

                    if similaridade_fake >= 0.87:
                        dic_resul = {}

                        lista_palavras_fake = []
                        if any(i in sent_encontrada_fake.text for i in exp_internacionais) and any(i in valor['Variavel'] for i in exp_internacionais):
                            for token_frase in remove_stop_words(valor['Tipo']):
                                if token_frase.text in sent_encontrada_fake.text:
                                    lista_palavras_fake.append(token_frase)

                            for token_var in remove_stop_words(valor['TipoV']):
                                if token_var.text in sent_encontrada_fake.text:
                                    lista_palavras_fake.append(token_var)
```

```
                        if (len(lista_palavras_fake) >= 2) and valor['Valor'] in valor_trecho_fake:
                            resposta_flag_fake = 0
                            resposta_flag_pbr = 1
                            id_novo += 1
                            url = item_fake['Url']
                            site_nome = urlparse(url).netloc
                            dic_resul['Id'] = id_novo
                            dic_resul['Site'] = site_nome
                            dic_resul['Url'] = url
                            dic_resul['Texto alvo'] = texto_completo_fake.text
                            dic_resul['Trecho destaque'] = sent_encontrada_fake.text
                            dic_resul['Valor destacado'] = valor_trecho_fake
                            dic_resul['Trecho encontrado'] = valor['Frase']
                            dic_resul['DataFrame respectivo'] = id_sidra
                            dic_resul['Resultado'] = 'Dado veridico'
                            resultados_pobreza.append(dic_resul)
                            break

                        elif (len(lista_palavras_fake) >= 2) and valor['Valor'] not in valor_trecho_fake:
                            resposta_flag_fake = 1
                            trecho_fake = valor['Frase']
                            sidra_fake = data['Id_Sidra']

                        if any(i in sent_encontrada_fake.text for i in exp_nacionais) and any(i in valor['Variavel'] for i in exp_nacionais):
                            for token_frase in remove_stop_words(valor['Tipo']):
                                if token_frase.text in sent_encontrada_fake.text:
                                    lista_palavras_fake.append(token_frase)

                            for token_var in remove_stop_words(valor['TipoV']):
                                if token_var.text in sent_encontrada_fake.text:
                                    lista_palavras_fake.append(token_var)

                            if (len(lista_palavras_fake) >= 2) and valor['Valor'] in valor_trecho_fake:
                                resposta_flag_pbr = 1
                                resposta_flag_fake = 0
                                id_novo += 1
                                url = item_fake['Url']
                                site_nome = urlparse(url).netloc
                                dic_resul['Id'] = id_novo
                                dic_resul['Site'] = site_nome
                                dic_resul['Url'] = url
                                dic_resul['Texto alvo'] = texto_completo_fake.text
                                dic_resul['Trecho destaque'] = sent_encontrada_fake.text
                                dic_resul['Valor destacado'] = valor_trecho_fake
                                dic_resul['Trecho encontrado'] = valor['Frase']
                                dic_resul['DataFrame respectivo'] = id_sidra
                                dic_resul['Resultado'] = 'Dado veridico'
                                resultados_pobreza.append(dic_resul)
                                break
```

```
    elif (len(lista_palavras_fake) >= 2) and valor['Valor'] not in valor_trecho_fake:
        resposta_flag_fake = 1
        trecho_fake = valor['Frase']
        sidra_fake = data['Id_Sidra']
else:
    for token_frase in remove_stop_words(valor['Tipo']):
        if token_frase.text in sent_encontrada_fake.text:
            lista_palavras_fake.append(token_frase)

    for token_var in remove_stop_words(valor['TipoV']):
        if token_var.text in sent_encontrada_fake.text:
            lista_palavras_fake.append(token_var)

    if (len(lista_palavras_fake) >= 2) and valor['Valor'] in valor_trecho_fake:
        resposta_flag_pbr = 1
        resposta_flag_fake = 0
        id_novo += 1
        url = item_fake['Url']
        site_nome = urlparse(url).netloc
        dic_resul['Id'] = id_novo
        dic_resul['Site'] = site_nome
        dic_resul['Url'] = url
        dic_resul['Texto alvo'] = texto_completo_fake.text
        dic_resul['Trecho destaque'] = sent_encontrada_fake.text
        dic_resul['Valor destacado'] = valor_trecho_fake
        dic_resul['Trecho encontrado'] = valor['Frase']
        dic_resul['DataFrame respectivo'] = id_sidra
        dic_resul['Resultado'] = 'Dado veridico'
        resultados_pobreza.append(dic_resul)
        break

    elif (len(lista_palavras_fake) >= 3) and valor['Valor'] not in valor_trecho_fake:
        resposta_flag_fake = 1
        trecho_fake = valor['Frase']
        sidra_fake = data['Id_Sidra']
```

```
if resposta_flag_fake == 1 and resposta_flag_pbr == 0 :
    dic_resul = {}
    resposta_flag_pbr = 1
    id_novo += 1
    url = item_fake['Url']
    site_nome = urlparse(url).netloc
    dic_resul['Id'] = id_novo
    dic_resul['Site'] = site_nome
    dic_resul['Url'] = url
    dic_resul['Texto alvo'] = texto_completo_fake.text
    dic_resul['Trecho destaque'] = sent_encontrada_fake.text
    dic_resul['Valor destacado'] = valor_trecho_fake
    dic_resul['Trecho encontrado'] = trecho_fake
    dic_resul['DataFrame respectivo'] = sidra_fake
    dic_resul['Resultado'] = 'Dado inveridico'
    resultados_pobreza.append(dic_resul)


elif resposta_flag_fake == 0 and resposta_flag_pbr == 0:
    dic_resul = {}
    id_novo += 1
    url = item_fake['Url']
    site_nome = urlparse(url).netloc
    dic_resul['Id'] = id_novo
    dic_resul['Site'] = site_nome
    dic_resul['Url'] = url
    dic_resul['Texto alvo'] = texto_completo_fake.text
    dic_resul['Trecho destaque'] = sent_encontrada_fake.text
    dic_resul['Valor destacado'] = valor_trecho_fake
    dic_resul['Trecho encontrado'] = None
    dic_resul['DataFrame respectivo'] = None
    dic_resul['Resultado'] = 'Imprecisão na validação, base de dados não suporta'
    resultados_pobreza.append(dic_resul)
```

Fonte: Elaborado pelo autor (2024).

3.5.2 Apenas respostas semelhantes ao ano

Interessando-se em um cenário a qual as buscas por todos os *DataFrames* de um assunto, não resultou em respostas semelhantes em ano e valor percentual, a análise ocorre de uma maneira semelhante à Figura 37 (com a exceção de certas peculiaridades para o caso da “Erradicação da pobreza).

O resultado proveniente das funções presentes na Figura 33, passa por uma busca procurando dados que possuem valores semelhantes quanto ao ano destacado. Novamente se faz útil o atributo “*loc*” da biblioteca pandas, buscando assim respostas em todos os *DataFrames* que correspondem ao mesmo valor do ano especificado pelo item. Caso a busca retorne uma resposta não vazia, os dados do item e suas respectivas respostas são salvas em uma nova lista de dicionários, a qual será a base para a análise.

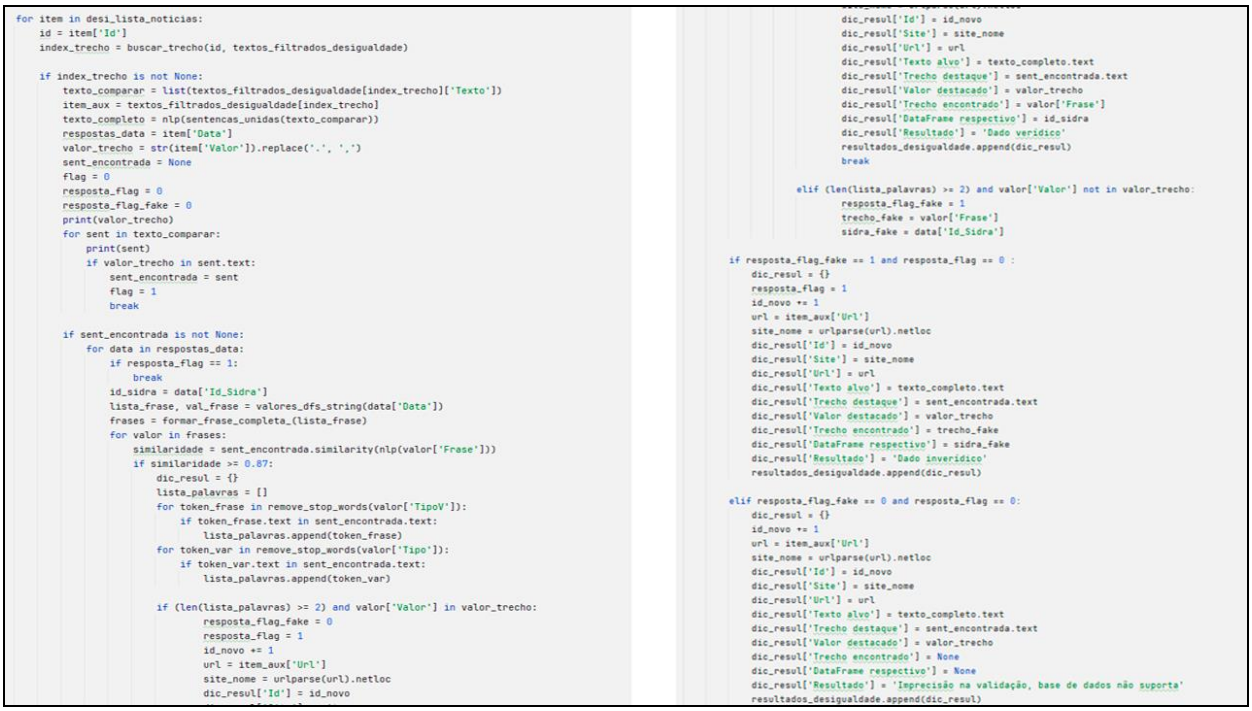
A seguir, a estrutura da etapa é esclarecida pela Figura 38 e 39.

Figura 38 - Organização dos dados semelhantes apenas ao ano. Assunto relacionado a desigualdades.

```
if len(iguais_desigualdade) == 0:
    desi_lista_noticias = []
    for item in comp_desigualdade:
        if 'Ano' in item and 'Dado' in item:
            dic = {}
            respostas = []
            dic['Id'] = item['Id']
            valor = item['Dado'].replace("%", "")
            valor = float(valor.replace(",", "."))
            ano_buscar = item['Ano']

            for data in sidra_desigualdades:
                dic_aux = {}
                id_sidra = data['Id']
                for data_a in data['Data']:
                    resposta = data_a['DataFrame'].loc[(data_a['DataFrame']['D2N - Ano'] == ano_buscar) & (
                                data_a['DataFrame']['MN - Unidade de Medida'] == '%')]
                    if not resposta.empty:
                        dic_aux['Id_Sidra'] = id_sidra
                        dic_aux['Data'] = resposta
                        respostas.append(dic_aux)

            if not len(respostas) == 0:
                dic['Valor'] = item['Dado']
                dic['Ano_valor'] = ano_buscar
                dic['Data'] = respostas
                desi_lista_noticias.append(dic)
```

Fonte: Elaborado pelo autor (2024).

Figura 39 - Análise para textos em torno do tema de desigualdades.



```
for item in desi_lista_noticias:
    id = item['Id']
    index_trecho = buscar_trecho(id, textos_filtrados_desigualdade)

    if index_trecho is not None:
        texto_comparar = list(textos_filtrados_desigualdade[index_trecho]['Texto'])
        item_aux = textos_filtrados_desigualdade[index_trecho]
        texto_completo = nlp(sentencas_unidas(texto_comparar))
        respostas_data = item['Data']
        valor_trecho = str(item['Valor']).replace('.', ',')
        sent_encontrada = None
        flag = 0
        resposta_flag = 0
        resposta_flag_fake = 0
        print(valor_trecho)
        for sent in texto_comparar:
            print(sent)
            if valor_trecho in sent.text:
                sent_encontrada = sent
                flag = 1
                break

        if sent_encontrada is not None:
            for data in respostas_data:
                if resposta_flag == 1:
                    break
                id_sidra = data['Id_Sidra']
                lista_frase, val_frase = valores_dfs_string(data['Data'])
                frases = formar_frase_completa_(lista_frase)
                for valor in frases:
                    similaridade = sent_encontrada.similarity(nlp(valor['Frase']))
                    if similaridade >= 0.87:
                        dic_resul = {}
                        lista_palavras = []
                        for token_frase in remove_stop_words(valor['TipoV']):
                            if token_frase.text in sent_encontrada.text:
                                lista_palavras.append(token_frase)
                        for token_var in remove_stop_words(valor['Tipo']):
                            if token_var.text in sent_encontrada.text:
                                lista_palavras.append(token_var)

                        if (len(lista_palavras) >= 2) and valor['Valor'] in valor_trecho:
                                resposta_flag_fake = 0
                                resposta_flag = 1
                                id_novo += 1
                                url = item_aux['Url']
                                site_nome = urlparse(url).netloc
                                dic_resul['Id'] = id_novo
                                dic_resul['Site'] = site_nome
                                dic_resul['Url'] = url
                                dic_resul['Texto alvo'] = texto_completo.text
                                dic_resul['Trecho destaque'] = sent_encontrada.text
                                dic_resul['Valor destacado'] = valor_trecho
                                dic_resul['Trecho encontrado'] = valor['Frase']
                                dic_resul['DataFrame respectivo'] = id_sidra
                                dic_resul['Resultado'] = 'Dado veridico'
                                resultados_desigualdade.append(dic_resul)
                                break

                        elif (len(lista_palavras) >= 2) and valor['Valor'] not in valor_trecho:
                                resposta_flag_fake = 1
                                trecho_fake = valor['Frase']
                                sidra_fake = data['Id_Sidra']

    if resposta_flag_fake == 1 and resposta_flag == 0 :
        dic_resul = {}
        resposta_flag = 1
        id_novo += 1
        url = item_aux['Url']
        site_nome = urlparse(url).netloc
        dic_resul['Id'] = id_novo
        dic_resul['Site'] = site_nome
        dic_resul['Url'] = url
        dic_resul['Texto alvo'] = texto_completo.text
        dic_resul['Trecho destaque'] = sent_encontrada.text
        dic_resul['Valor destacado'] = valor_trecho
        dic_resul['Trecho encontrado'] = trecho_fake
        dic_resul['DataFrame respectivo'] = sidra_fake
        dic_resul['Resultado'] = 'Dado inveridico'
        resultados_desigualdade.append(dic_resul)

    elif resposta_flag_fake == 0 and resposta_flag == 0:
        dic_resul = {}
        id_novo += 1
        url = item_aux['Url']
        site_nome = urlparse(url).netloc
        dic_resul['Id'] = id_novo
        dic_resul['Site'] = site_nome
        dic_resul['Url'] = url
        dic_resul['Texto alvo'] = texto_completo.text
        dic_resul['Trecho destaque'] = sent_encontrada.text
        dic_resul['Valor destacado'] = valor_trecho
        dic_resul['Trecho encontrado'] = None
        dic_resul['DataFrame respectivo'] = None
        dic_resul['Resultado'] = 'Imprecisão na validação, base de dados não suporta'
        resultados_desigualdade.append(dic_resul)
```

Fonte: Elaborado pelo autor (2024).

4 RESULTADOS

Com a metodologia apresentada e detalhada, interessa-se avliar os resultados obtidos a partir da prática escolhida.

O tópico irá abranger resultados quanto aos textos noticiários de cada assunto e também valores provenientes de textos gerados especificamente para o projeto.

4.1 Textos noticiários

As análises de forma geral causaram um resultado insatisfatório, não se distanciando do esperado durante as últimas etapas do projeto.

Quadro 2 - Resultado verídico “Erradicação da pobreza”.

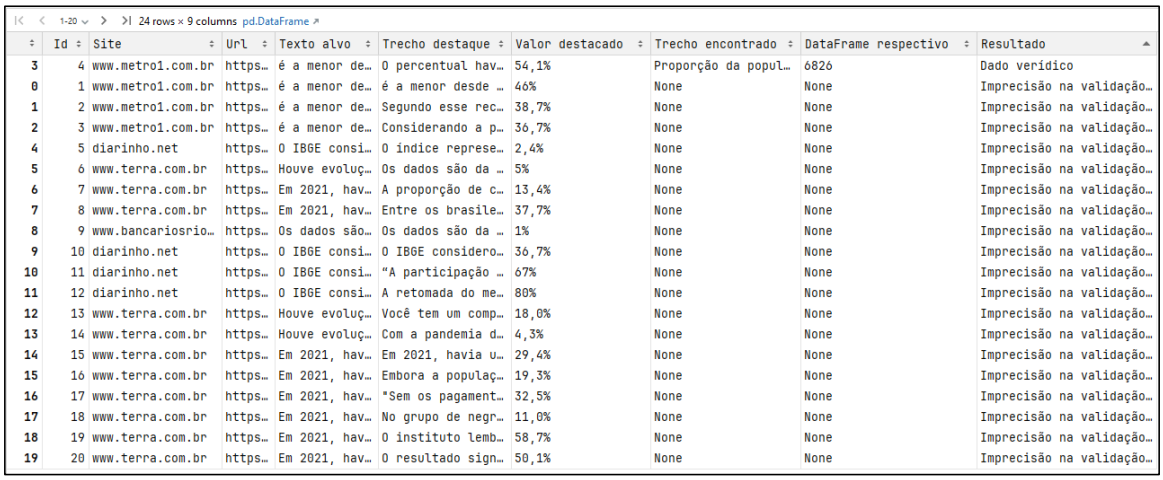
Texto destaque	Valor destacado	Texto encontrado
“Quinta-feira, 23 de maio de 2024Home/Notícias/Brasil/IBGE divulga que quase metade das crianças até 5 anos vivia em situação de pobreza no Brasil em 2022. Percentual havia sido ainda mais elevado em 2021, quando 54,1% das crianças de zero a cinco anos estavam em situação de pobreza, patamar recorde da série histórica iniciada em 2012Foto: Rovena Rosa/Agência BrasilPor: Metro1 no dia 09 de abril de 2024 às 20:16O Instituto Brasileiro de Geografia e Estatística (IBGE) divulgou, nesta terça-feira (9), dados apontando que quase metade das crianças de zero a cinco anos vivia em situação de pobreza no Brasil em 2022.”	54,1%	Proporção da população abaixo da linha de pobreza nacional é dado(a) por 54.1%, a partir do(a) grupo de idade 0 a 5 anos no Brasil e no ano de 2021

Fonte: Elaborado pelo autor (2024).

Considerando todos os assuntos definidos pelo projeto, apenas foi possível apontar a veridicidade de uma notícia encontrada no tema “Erradicação da pobreza”. O portal de notícias destacado foi o “metro1”, o trecho a qual cita o dado verificado e o texto encontrado a partir da base de dados do IBGE, pode ser visualizado no Quadro 2.

O *DataFrame* a qual estampa as respostas obtidas pelas análises das notícias quanto “Erradicação da pobreza” se mostra presente na Figura 41.

Figura 40 - *DataFrame* análise textos noticiários tema “Erradicação da pobreza”.



1-20 24 rows × 9 columns pd.DataFrame

	Id	Site	Url	Texto alvo	Trecho destaque	Valor destacado	Trecho encontrado	DataFrame respectivo	Resultado
3	4	www.metro1.com.br	https…	é a menor de…	O percentual hav…	54,1%	Proporção da popul…	6826	Dado verídico
0	1	www.metro1.com.br	https…	é a menor de…	é a menor desde …	46%	None	None	Imprecisão na validação…
1	2	www.metro1.com.br	https…	é a menor de…	Segundo esse rec…	38,7%	None	None	Imprecisão na validação…
2	3	www.metro1.com.br	https…	é a menor de…	Considerando a p…	36,7%	None	None	Imprecisão na validação…
4	5	diarinho.net	https…	O IBGE consi…	O índice represe…	2,4%	None	None	Imprecisão na validação…
5	6	www.terra.com.br	https…	Houve evoluç…	Os dados são da …	5%	None	None	Imprecisão na validação…
6	7	www.terra.com.br	https…	Em 2021, hav…	A proporção de c…	13,4%	None	None	Imprecisão na validação…
7	8	www.terra.com.br	https…	Em 2021, hav…	Entre os brasile…	37,7%	None	None	Imprecisão na validação…
8	9	www.bancariosrio…	https…	Os dados são…	Os dados são da …	1%	None	None	Imprecisão na validação…
9	10	diarinho.net	https…	O IBGE consi…	O IBGE considero…	36,7%	None	None	Imprecisão na validação…
10	11	diarinho.net	https…	O IBGE consi…	“A participação …	67%	None	None	Imprecisão na validação…
11	12	diarinho.net	https…	O IBGE consi…	A retomada do me…	80%	None	None	Imprecisão na validação…
12	13	www.terra.com.br	https…	Houve evoluç…	Você tem um comp…	18,0%	None	None	Imprecisão na validação…
13	14	www.terra.com.br	https…	Houve evoluç…	Com a pandemia d…	4,3%	None	None	Imprecisão na validação…
14	15	www.terra.com.br	https…	Em 2021, hav…	Em 2021, havia u…	29,4%	None	None	Imprecisão na validação…
15	16	www.terra.com.br	https…	Em 2021, hav…	Embora a populaç…	19,3%	None	None	Imprecisão na validação…
16	17	www.terra.com.br	https…	Em 2021, hav…	"Sem os pagament…	32,5%	None	None	Imprecisão na validação…
17	18	www.terra.com.br	https…	Em 2021, hav…	No grupo de negr…	11,0%	None	None	Imprecisão na validação…
18	19	www.terra.com.br	https…	Em 2021, hav…	O instituto lemb…	58,7%	None	None	Imprecisão na validação…
19	20	www.terra.com.br	https…	Em 2021, hav…	O resultado sign…	50,1%	None	None	Imprecisão na validação…

Fonte: Elaborado pelo autor (2024).

Todos *DataFrames* receberam a mesma organização:

- Id: cada item da tabela recebeu um novo Id para melhor organização;
- Site: descreve o nome do site a qual o texto foi retirado;
- Texto alvo: apresenta todo o texto especificado para análise;
- Trecho destaque: define o trecho específico a qual o valor percentual e seu respectivo ano foi encontrado;
- Valor destaque: representa o valor encontrado no texto;
- Trecho encontrado: retorna uma frase a qual reúne as especificações do dado encontrado na base do IBGE do assunto relacionado;

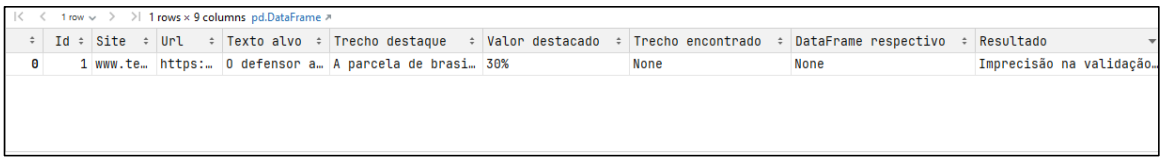
- *DataFrame* respectivo: representa o Id do *DataFrame* da base de dados verídicos (proveniente da API do IBGE), facilitando uma futura verificação;
- Resultado: define o resultado de todo o processo de análise.

O restante do *DataFrame* apresentou imprecisão na realização da análise. Apesar de ser um resultado indesejado, o mesmo não deixa de ser coerente. Parte dos trechos apresentaram um contexto semelhante ao armazenado pela base de bases, entretanto havia discrepâncias em torno principalmente do grupo pesquisado ou a da própria omissão do mesmo.

Outro fator determinante para tal resultado, está relacionado diretamente com a falta de determinado indicador de pesquisa dentro dos dados armazenados pela API do IBGE. Dessa forma o texto apresenta uma alta similaridade, por estar tratando sobre a taxa de pobreza no Brasil, contudo a variável não se mostra a mesma. Algo que também pode ocorrer quando se trata de determinado ano que a base de dados do IBGE não possui. Dessa maneira, mesmo o assunto e a variável sendo coerente entre o texto e a base de dados, se torna uma análise imprecisa já que não acontece a relação correta quanto ao ano. Sendo incorreto afirmar uma resposta quanto a veridicidade.

Quanto ao assunto “Furto e roubo“: realizar pesquisas que se assemelham aos dados especificados pela base de dados verídicos, proveniente da API do IBGE, resultou em uma quantidade de apenas um *url* válido. A etapa de análise também refletiu imprecisão. A Figura 42 representa tal saída.

Figura 41 - *DataFrame* análise textos noticiários tema “Furto e roubo”.



1 row 1 rows × 9 columns pd.DataFrame

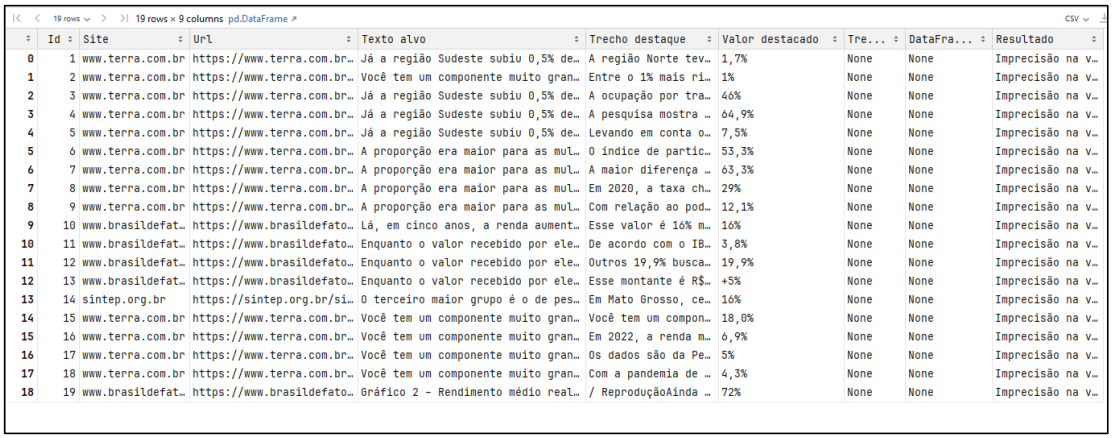
	Id	Site	Url	Texto alvo	Trecho destaque	Valor destacado	Trecho encontrado	DataFrame respectivo	Resultado
0	1	www.te…	https:…	O defensor a…	A parcela de brasi…	30%	None	None	Imprecisão na validação…

Fonte: Elaborado pelo autor (2024).

Quanto a “Redução das desigualdades” foi possível extrair de uma forma efetiva tanto *links* quanto seus respectivos dados presentes em seus trechos. Contudo, o resultado da análise continuou ressaltando a imprecisão, pelos

mesmos motivos discutidos anteriormente. O resultado está presente na Figura 43.

Figura 42 - *DataFrame* análise textos noticiários tema “Redução das desigualdades”.



19 rows × 9 columns pd.DataFrame ↗ CSV

	Id	Site	Url	Texto alvo	Trecho destaque	Valor destacado	Tre...	DataFra...	Resultado
0	1	www.terra.com.br	https://www.terra.com.br…	Já a região Sudeste subiu 0,5% de…	A região Norte tev…	1,7%	None	None	Imprecisão na v…
1	2	www.terra.com.br	https://www.terra.com.br…	Você tem um componente muito gran…	Entre o 1% mais ri…	1%	None	None	Imprecisão na v…
2	3	www.terra.com.br	https://www.terra.com.br…	Já a região Sudeste subiu 0,5% de…	A ocupação por tra…	46%	None	None	Imprecisão na v…
3	4	www.terra.com.br	https://www.terra.com.br…	Já a região Sudeste subiu 0,5% de…	A pesquisa mostra …	64,9%	None	None	Imprecisão na v…
4	5	www.terra.com.br	https://www.terra.com.br…	Já a região Sudeste subiu 0,5% de…	Levando em conta o…	7,5%	None	None	Imprecisão na v…
5	6	www.terra.com.br	https://www.terra.com.br…	A proporção era maior para as mul…	O índice de partic…	53,3%	None	None	Imprecisão na v…
6	7	www.terra.com.br	https://www.terra.com.br…	A proporção era maior para as mul…	A maior diferença …	63,3%	None	None	Imprecisão na v…
7	8	www.terra.com.br	https://www.terra.com.br…	A proporção era maior para as mul…	Em 2020, a taxa ch…	29%	None	None	Imprecisão na v…
8	9	www.terra.com.br	https://www.terra.com.br…	A proporção era maior para as mul…	Com relação ao pod…	12,1%	None	None	Imprecisão na v…
9	10	www.brasildefat…	https://www.brasildefato…	Lá, em cinco anos, a renda aument…	Esse valor é 16% m…	16%	None	None	Imprecisão na v…
10	11	www.brasildefat…	https://www.brasildefato…	Enquanto o valor recebido por ele…	De acordo com o IB…	3,8%	None	None	Imprecisão na v…
11	12	www.brasildefat…	https://www.brasildefato…	Enquanto o valor recebido por ele…	Outros 19,9% busca…	19,9%	None	None	Imprecisão na v…
12	13	www.brasildefat…	https://www.brasildefato…	Enquanto o valor recebido por ele…	Esse montante é R$…	+5%	None	None	Imprecisão na v…
13	14	sintep.org.br	https://sintep.org.br/si…	O terceiro maior grupo é o de pes…	Em Mato Grosso, ce…	16%	None	None	Imprecisão na v…
14	15	www.terra.com.br	https://www.terra.com.br…	Você tem um componente muito gran…	Você tem um compon…	18,0%	None	None	Imprecisão na v…
15	16	www.terra.com.br	https://www.terra.com.br…	Você tem um componente muito gran…	Em 2022, a renda m…	6,9%	None	None	Imprecisão na v…
16	17	www.terra.com.br	https://www.terra.com.br…	Você tem um componente muito gran…	Os dados são da Pe…	5%	None	None	Imprecisão na v…
17	18	www.terra.com.br	https://www.terra.com.br…	Você tem um componente muito gran…	Com a pandemia de …	4,3%	None	None	Imprecisão na v…
18	19	www.brasildefat…	https://www.brasildefato…	Gráfico 2 - Rendimento médio real…	/ ReproduçãoAinda …	72%	None	None	Imprecisão na v…

Fonte: Elaborado pelo autor (2024).

Os assuntos “Saúde” e “Meio Ambiente” apresentaram um resultado inusitado em comparação aos demais. Diferentemente dos casos anteriores, todos os links provenientes da API de busca em torno do *GoogleNews,* foram classificados com inválidos (página não encontrada). O que impossibilitou a posterior análise.

4.2 Textos de linguagem natural aleatórios

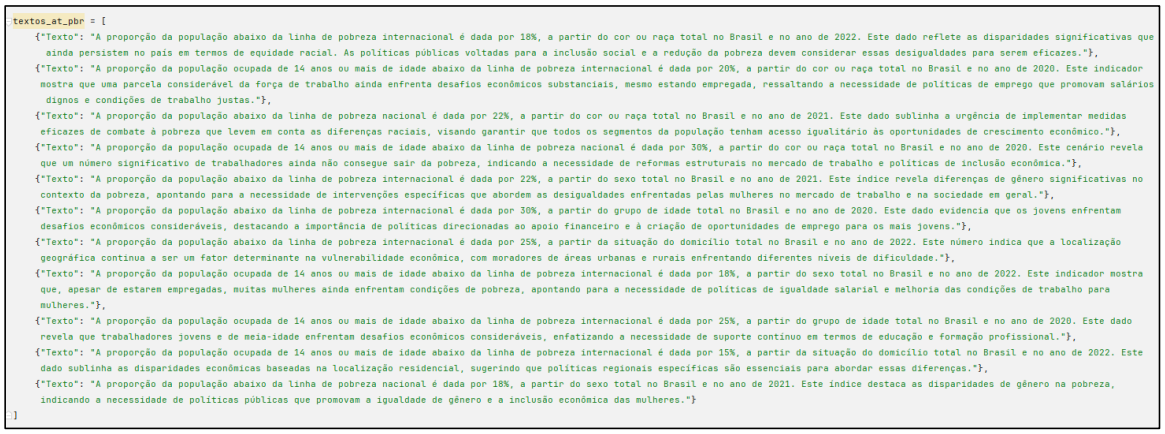
Vislumbrar a ineficácia perante os textos noticiários, efeito de uma base de dados limitada, não confirma a falta de praticidade do método para textos que se enquadram perante aos assuntos pesquisados.

Buscando tal validação, foram gerados textos de forma aleatória, se beneficiando da facilidade e aleatoriedade gerada pelo uso do *Chat GPT*, a partir da base de cada identificador presente em cada assunto.

O caso “Erradicação da pobreza” foi selecionado para demonstrar a possibilidade criada pela nova proposta. Logo, a Figura 44 imprimi os textos gerados para o tema. Como no contexto atual não se faz necessário o armazenamento de *url*, o armazenamento se faz de uma forma mais simples.

Dessa forma, os textos foram organizados em uma lista de dicionários com chave única “Texto”.

Figura 43 - Textos aleatórios gerados sobre “Erradicação da pobreza”.

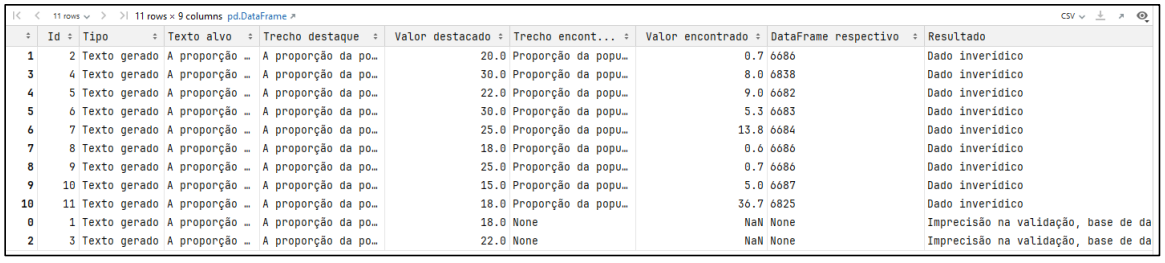


```
textos_at_pbr = [
    {"Texto": "A proporção da população abaixo da linha de pobreza internacional é dada por 18%, a partir do cor ou raça total no Brasil e no ano de 2022. Este dado reflete as disparidades significativas que
      ainda persistem no país em termos de equidade racial. As políticas públicas voltadas para a inclusão social e a redução da pobreza devem considerar essas desigualdades para serem eficazes."},
    {"Texto": "A proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional é dada por 20%, a partir do cor ou raça total no Brasil e no ano de 2020. Este indicador
     mostra que uma parcela considerável da força de trabalho ainda enfrenta desafios econômicos substanciais, mesmo estando empregada, ressaltando a necessidade de políticas de emprego que promovam salários
      dignos e condições de trabalho justas."},
    {"Texto": "A proporção da população abaixo da linha de pobreza nacional é dada por 22%, a partir do cor ou raça total no Brasil e no ano de 2021. Este dado sublinha a urgência de implementar medidas
     eficazes de combate à pobreza que levem em conta as diferenças raciais, visando garantir que todos os segmentos da população tenham acesso igualitário às oportunidades de crescimento econômico."},
    {"Texto": "A proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza nacional é dada por 30%, a partir do cor ou raça total no Brasil e no ano de 2020. Este cenário revela
     que um número significativo de trabalhadores ainda não consegue sair da pobreza, indicando a necessidade de reformas estruturais no mercado de trabalho e políticas de inclusão econômica."},
    {"Texto": "A proporção da população abaixo da linha de pobreza internacional é dada por 22%, a partir do sexo total no Brasil e no ano de 2021. Este índice revela diferenças de gênero significativas no
     contexto da pobreza, apontando para a necessidade de intervenções específicas que abordem as desigualdades enfrentadas pelas mulheres no mercado de trabalho e na sociedade em geral."},
    {"Texto": "A proporção da população abaixo da linha de pobreza internacional é dada por 30%, a partir do grupo de idade total no Brasil e no ano de 2020. Este dado evidencia que os jovens enfrentam
     desafios econômicos consideráveis, destacando a importância de políticas direcionadas ao apoio financeiro e à criação de oportunidades de emprego para os mais jovens."},
    {"Texto": "A proporção da população abaixo da linha de pobreza internacional é dada por 25%, a partir da situação do domicílio total no Brasil e no ano de 2022. Este número indica que a localização
     geográfica continua a ser um fator determinante na vulnerabilidade econômica, com moradores de áreas urbanas e rurais enfrentando diferentes níveis de dificuldade."},
    {"Texto": "A proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional é dada por 18%, a partir do sexo total no Brasil e no ano de 2022. Este indicador mostra
     que, apesar de estarem empregadas, muitas mulheres ainda enfrentam condições de pobreza, apontando para a necessidade de políticas de igualdade salarial e melhoria das condições de trabalho para
     mulheres."},
    {"Texto": "A proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional é dada por 25%, a partir do grupo de idade total no Brasil e no ano de 2020. Este dado
     revela que trabalhadores jovens e de meia-idade enfrentam desafios econômicos consideráveis, enfatizando a necessidade de suporte contínuo em termos de educação e formação profissional."},
    {"Texto": "A proporção da população ocupada de 14 anos ou mais de idade abaixo da linha de pobreza internacional é dada por 15%, a partir da situação do domicílio total no Brasil e no ano de 2022. Este
     dado sublinha as disparidades econômicas baseadas na localização residencial, sugerindo que políticas regionais específicas são essenciais para abordar essas diferenças."},
    {"Texto": "A proporção da população abaixo da linha de pobreza nacional é dada por 18%, a partir do sexo total no Brasil e no ano de 2021. Este índice destaca as disparidades de gênero na pobreza,
     indicando a necessidade de políticas públicas que promovam a igualdade de gênero e a inclusão econômica das mulheres."}
]
```

Fonte: Elaborado pelo autor (2024).

Considerando o tema presente na Figura 44, os resultados da análise foram factualmente satisfatórios, ao passo que foi possível identificar dados inverídicos. Mesmo assim, ainda foi presente alguns poucos resultados de imprecisão, isso se deve pela ausência da relação do ano entre as partes. Os resultados são estampados na Figura 45, sendo possível certas mudanças na organização da tabela: por não se tratar de textos provenientes de sites, se torna irrelevante criar colunas para retornar o nome do site e seu respectivo *url*. Logo, as mesmas foram substituídas para a coluna “Tipo”, a qual ressalta que é um texto gerado aleatoriamente para o contexto do projeto. Outra mudança está presente no acréscimo da coluna “Valor encontrado” que retorna o valor encontrado na base de dados relacionada a busca.

Figura 44 - *DataFrame* resultado da análise para o assunto “Erradicação da pobreza”.



11 rows × 9 columns pd.DataFrame ↗ CSV

	Id	Tipo	Texto alvo	Trecho destaque	Valor destacado	Trecho encont...	Valor encontrado	DataFrame respectivo	Resultado
1	2	Texto gerado	A proporção …	A proporção da po…	20.0	Proporção da popu…	0.7	6686	Dado inverídico
3	4	Texto gerado	A proporção …	A proporção da po…	30.0	Proporção da popu…	8.0	6838	Dado inverídico
4	5	Texto gerado	A proporção …	A proporção da po…	22.0	Proporção da popu…	9.0	6682	Dado inverídico
5	6	Texto gerado	A proporção …	A proporção da po…	30.0	Proporção da popu…	5.3	6683	Dado inverídico
6	7	Texto gerado	A proporção …	A proporção da po…	25.0	Proporção da popu…	13.8	6684	Dado inverídico
7	8	Texto gerado	A proporção …	A proporção da po…	18.0	Proporção da popu…	0.6	6686	Dado inverídico
8	9	Texto gerado	A proporção …	A proporção da po…	25.0	Proporção da popu…	0.7	6686	Dado inverídico
9	10	Texto gerado	A proporção …	A proporção da po…	15.0	Proporção da popu…	5.0	6687	Dado inverídico
10	11	Texto gerado	A proporção …	A proporção da po…	18.0	Proporção da popu…	36.7	6825	Dado inverídico
0	1	Texto gerado	A proporção …	A proporção da po…	18.0	None	NaN	None	Imprecisão na validação, base de da
2	3	Texto gerado	A proporção …	A proporção da po…	22.0	None	NaN	None	Imprecisão na validação, base de da

Fonte: Elaborado pelo autor (2024).

O Quadro 3 evidencia alguns exemplos identificados de textos com dados inverídicos e o respectivo dado correspondente presente na base de dados proveniente do IBGE.

Quadro 3 - Resultados inverídicos de textos gerados sobre “Erradicação da pobreza”

Texto destaque	**Valor destacado**	**Texto encontrado**
A proporção da população abaixo da linha de pobreza internacional é dada por 22%, a partir do sexo total no Brasil e no ano de 2021.	22%	Proporção da população abaixo da linha de pobreza internacional é dado(a) por 9.0%, a partir do(a) sexo total no Brasil e no ano de 2021.
A proporção da população abaixo da linha de pobreza internacional é dada por 30%, a partir do grupo de idade total no Brasil e no ano de 2020.	30%	Proporção da população abaixo da linha de pobreza internacional é dado(a) por 6.0%, a partir do(a) grupo de idade total no Brasil e no ano de 2020.
A proporção da população abaixo da linha de pobreza nacional é dada por 18%, a partir do sexo total no Brasil e no ano de 2021.	18%	Proporção da população abaixo da linha de pobreza nacional é dado(a) por 36.7%, a partir do(a) sexo total no Brasil e no ano de 2021.

Fonte: Elaborado pelo autor (2024).

Com os resultados satisfatórios se faz possível utilizar outros conhecimentos em torno da ciência de dados, tornando se mais visual a análise.

Utilizando a linguagem *Python* juntamente com a biblioteca “*matplotlib*” se faz possível a fácil criação de diferentes formatos de gráficos. A exploração de dados por meio de gráficos e/ou tabelas, auxilia pontualmente na concretização de um novo conhecimento. O código abaixo (Figura 46) apresenta a plotagem de

um gráfico de pizza que visa identificar a quantidade e tipo de resultados obtidos pela análise descrita anteriormente, sendo ajustado os tamanhos da fonte e também o formato do gráfico.

Figura 45 - Código gráfico de pizza para análise proposta.

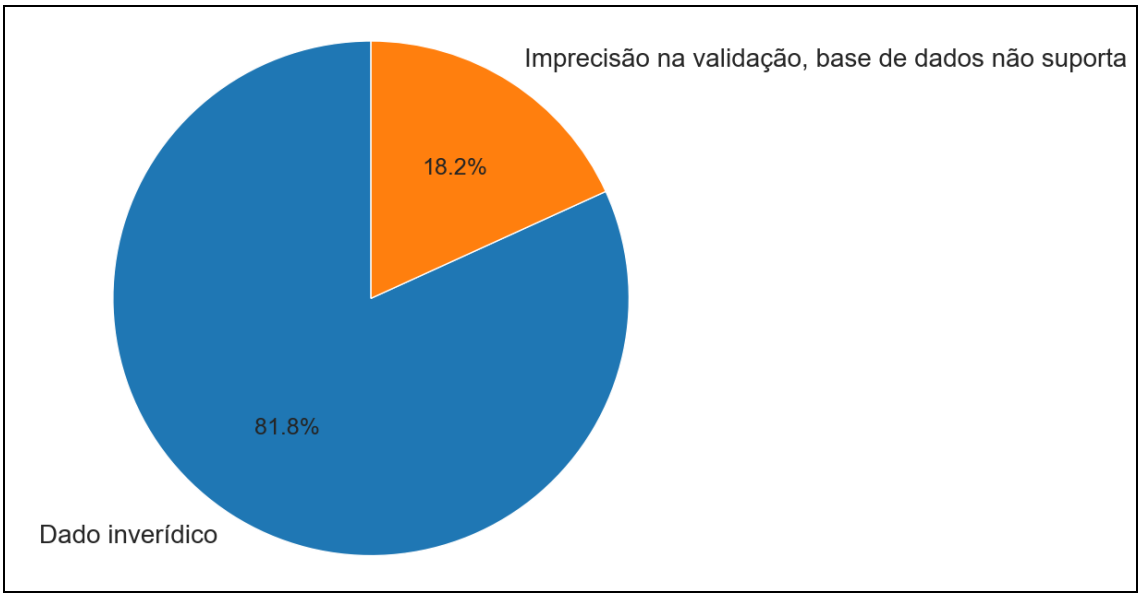
```
contagem_valores = df_pobreza_att['Resultado'].value_counts()
pedaco_val = contagem_valores.values
rotulos = contagem_valores.index
fig, ax = plt.subplots(figsize=(10, 8))

wedges, texts, autotexts = ax.pie(
    pedaco_val,
    labels=rotulos,
    autopct='%1.1f%%',
    startangle=90,
    textprops=dict(fontsize=20)
)



for autotext in autotexts:
    autotext.set_fontsize(18)

ax.axis('equal')
plt.show()
```

Fonte: Elaborado pelo autor (2024).

O resultado da codificação se mostra estampado na Figura 47. Apesar da simplicidade do gráfico, o mesmo revela de forma intuitiva e rápida a quantidade dos tipos de resultados presente no *DataFrame* gerado após a realização da análise. Tornando se claro que as quantidades de valores imprecisos se tornaram uma exceção e não o padrão, como anteriormente.

Outra relação de extrema relevância se mostra a partir da relação do valor destacado pelo texto analisado, ou seja, o valor presente no texto, com o respectivo valor condizente e presente na base de dados do tema analisado. Sendo assim, é possível visualizar o quanto um texto distorceu o dado verídico, observando seu valor real e o valor presente no texto.

Figura 46 - Gráfico de pizza, feito em *Python*, dos resultados da análise proposta.



Fonte: Elaborado pelo autor (2024).

Visando a concretização dessa relação entre valores, o gráfico de linhas foi escolhido. A Figura 48 demonstra a codificação necessária para a realização do mesmo.

Figura 47 - Código gráfico de linhas da relação dos valores de cada Id.

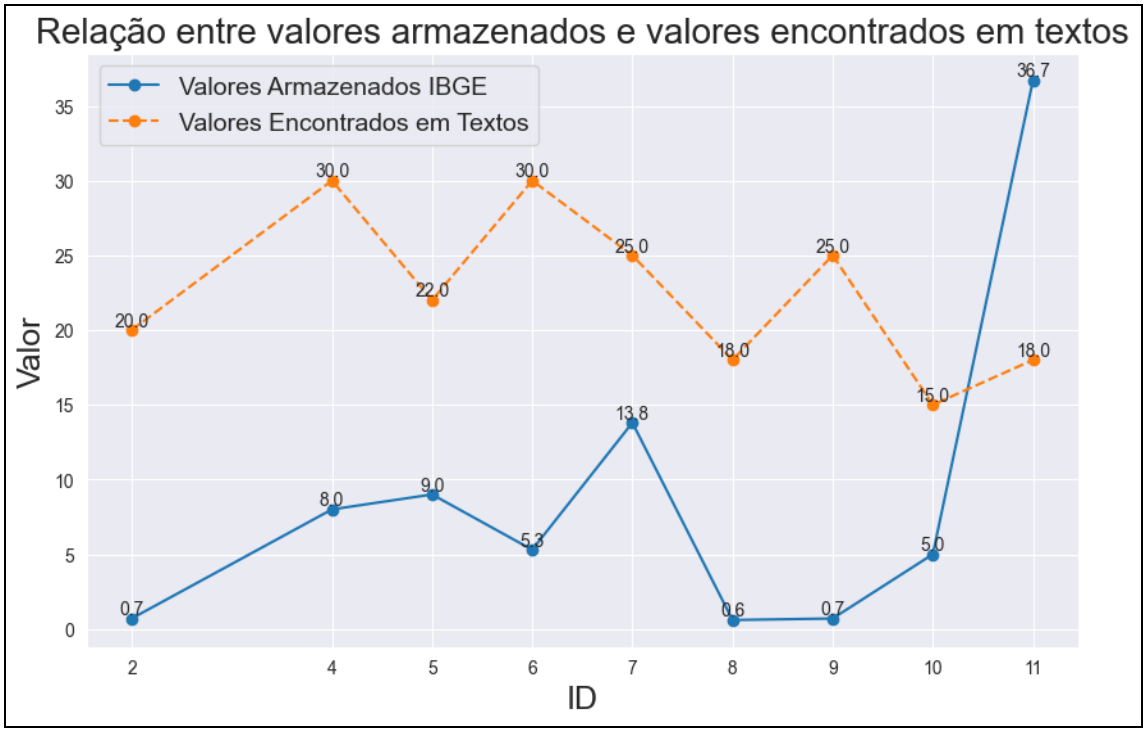
```
relacoes_valores = df_pobreza_att.loc[df_pobreza_att['Valor encontrado'].notna()]
indices = relacoes_valores['Id'].to_numpy()
valores_armazenados = relacoes_valores['Valor encontrado'].to_numpy()
valores_encontrados = relacoes_valores['Valor destacado'].to_numpy()

plt.figure(figsize=(10, 6))
plt.plot(indices, valores_armazenados, label='Valores Armazenados IBGE', marker='o', linestyle='-')
plt.plot(indices, valores_encontrados, label='Valores Encontrados em Textos', marker='o', linestyle='--')
plt.xlabel('ID', fontsize=18)
plt.ylabel('Valor', fontsize=18)
plt.title('Relação entre valores armazenados e valores encontrados em textos', fontsize=20)
plt.legend(fontsize=14)
plt.grid(True)

plt.xticks(indices[::1])
for i in range(len(indices)):
    plt.text(indices[i], valores_armazenados[i], str(valores_armazenados[i]), ha='center', va='bottom')
    plt.text(indices[i], valores_encontrados[i], str(valores_encontrados[i]), ha='center', va='bottom')

plt.show()
```

Fonte: Elaborado pelo autor (2024).

Inicialmente a tabela, resultante da análise, é filtrada para excluir casos imprecisos (valores nulos quanto a valores encontrados na base de dados verídicos). Adiante, os identificadores dos itens da tabela são postos como valores do eixo x, dessa forma o gráfico irá organizar os valore por Id. O código ainda abrange certas configurações para aumento das fontes e uma maior legibilidade do gráfico.

O resultado da execução do código se mostra presente na Figura 49.

Figura 48 - Gráfico de linhas, feito em *Python*, dos valores encontrados nos textos e valores armazenados na base de dados.



Fonte: Elaborado pelo autor (2024).

Observando o gráfico pode se verificar de forma intuitiva o quanto os textos analisados distorceram de forma discrepante dados reais. Mostrando assim a importância de sua realização, já que a partir do mesmo pode ser possível gerar um padrão (uma margem) de alterações de dados verídicos para a fabricação de uma narrativa inverídica.

Apesar do exemplo pontual para o tema em torno da pobreza, a mesma experimentação também se vê bem-sucedida para os demais assuntos apontados

durante o trabalho. Evidenciando que em dados presentes no cenário condizente a base de dados possui uma alta chance de ser tratado da maneira correta.

5 CONCLUSÃO

Discussões em torno da problemática de falsas notícias ainda apresenta uma longa estimativa de vida, sendo ainda possível vislumbrar seu funcionamento no ano presente de 2024. A veridicidade de fatos já estudados e organizados, deve ser mantida evitando invalidação de todo um trabalho científico ou até mesmo da negligência de assuntos.

Voltar os olhos pela concretização do trabalho vigente, também é se atentar com as dificuldades encontradas durante todo o processo.

O primeiro grande obstáculo girou em torno de uma filtragem coerente para a identificação de diversos dados em um texto. A primeira solução visava extrair o menor texto possível a partir de regras gerais, coletando apenas substantivos, pronomes, adjetivos, símbolos, números e outros sinais (Figura 50 demonstra a função responsável pela atividade). A partir de testes, foi identificado a falha de obter trechos válidos exigindo alteração no método. Alcançando a opção em torno da similaridade dos temas armazenados (apresentada no capítulo de metodologia).

Figura 49 - Função filtragem tipo de palavras.

```
def filtrar_palavras(setenca):
    doc = nlp(setenca)
    # Lista de categorias a serem mantidas
    categorias_mantidas = ["NOUN", "PROPN", "ADJ", "SYM", "PUNCT", "NUM"]
    tokens_filtrados = [token.text for token in doc if token.pos_ in categorias_mantidas]
    return nlp(" ".join(tokens_filtrados))
```

Fonte: Elaborado pelo autor (2024).

A escolha da facilidade encontrada pela API *GoogleNews* ao decorrer do projeto apresentou obstáculos na manutenção de uma boa exatidão. O fato do mesmo não ter alcançado resultados positivos para 2 (“Saúde” e “Meio Ambiente”) dos 5 temas selecionados, exibe pontualmente o fato. Logo, apesar da ferramenta proporcionar buscas bem-sucedidas a atenção novamente deve ser retomada para o estudo de um novo meio de coleta de notícias.

A análise em torno das notícias se mostrou insatisfatória pela falta de resultados precisos. Entretanto, o fato não se relaciona diretamente com erros encontrados durante a codificação, e sim da falta de dados suficientes para

construir um cenário robusto. O fato de muitos textos noticiários possuírem temas semelhantes, mas ao mesmo tempo distantes dos já armazenados (do IBGE) revela o principal motivo. Em suma, a API de agregados resulta uma base de dados sólida, mas que ainda estampa carência de atualizações e acréscimo de variáveis para cada tema, além da falta de informações importantes em sua documentação como é o caso da própria especificação do uso máximo da aplicação até ocorrer um caso de bloqueio temporário.

A idealização de alcançar meios para se analisar dados presentes em textos, pôde em partes ser concretizada. O bom funcionamento na verificação de dados verídicos ou inverídicos em textos com estruturação de maior semelhança com os temas armazenados, levanta a discussão e a confirmação da possibilidade de implementação de sistemas que realizam constantemente verificações da integridade de dados já pesquisados por estudiosos.

Dessa maneira, a abstração de formas para combater as famigeradas *fake news* se torna mais fraca. Estampando que apesar das dificuldades e limitações, há a possibilidade de criar cada vez mais um ambiente seguro para informações.

O caminho idealizado apresenta diversas lacunas a serem exploradas, e com possíveis melhoras a idealização de algo anteriormente distante, pode se tornar cada vez mais concreto. O material de coleta para análise pode ser mudado para uma nova ferramenta ou até mesmo a criação própria de novos métodos para sua coleta, corrigindo erros presentes na opção escolhida do trabalho. Voltar a atenção para a coleta de textos em redes sociais, pode ser uma maneira eficaz de aumentar a quantidade de material a ser analisado.

Além disso, há a necessidade de realizar um aumento da base de dados tratados como verdadeiros. A expansão de fontes confiáveis para a busca de dados verídicos, irá impactar diretamente na criação de um ambiente com maior assertividade para com suas análises. Utilizar ferramentas com capacidade de atualizar informações não atualizadas, sendo necessário até mesmo uma verificação manual de informações, irá possibilitar um cenário cada vez mais consistente.

O algoritmo escolhido também pode ser constantemente aperfeiçoado, sendo capaz de utilizar diversas técnicas, bibliotecas (o trabalho vigente utilizou apenas uma biblioteca em torno do PLN, ainda possuem outras alternativas), a qual não se fez presente durante o projeto. O conselho fica a margem da

implementação de um modelo próprio para interpretação de textos, focando em contornar os obstáculos antes enfrentados.

REFERÊNCIAS

1.1 Definitions. Government Of Canada, S. C. Disponível em: https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch1/definitions/5214853-eng.htm . Acesso em: 23 out. 2023

ALTARES, G. **A longa história das notícias falsas**. EL País, 2018. Disponível em: https://brasil.elpais.com/brasil/2018/06/08/cultura/1528467298_389944.html. Acesso em: 12 out. 2023.

ANTIĆ, Zhenya. **Python natural language processing cookbook: over 50 recipes to understand, analyze, and generate text for implementing language processing tasks**. London, England: Packt Publishing, Limited, 2021.

Aplicação de Ciência de Dados em diferentes áreas de negócios. HupData. Disponível em: https://hupdata.com/ciencia-de-dados-em-diferentes-setores-economicos/ . Acesso em: 8 out. 2023.

BIRD, S *et al* **Natural Language Processing with Python**. O'Reilly Media, Inc., 2009. Acesso em: 12 out. 2023.

CARVALHO, Caroline**. O que é Python? Um Guia para iniciar na Linguagem**. Alura, 2023. Disponível em: https://www.alura.com.br/artigos/python . Acesso em: 8 nov. 2023.

Chefe da Organização Mundial da Saúde declara o fim da COVID-19 como uma emergência de saúde global. Nações Unidas Brasil, 2023. Disponível em: https://brasil.un.org/pt-br/230307-chefe-da-organização-mundial-da-saúde-declara-o-fim-da-covid-19-como-uma-emergência-de-saúde . Acesso em: 11 out. 2023.

COMPROVA, P. **É enganoso que estudo de Harvard comprovou eficácia de hidroxicloroquina**. CNN Brasil, 2022. Disponível em:

https://www.cnnbrasil.com.br/nacional/e-falso-que-estudo-de-harvard-comprovou-eficacia-de-hidroxicloroquina/ . Acesso em: 12 out. 2023.

DA SILVA, Leandro Augusto; PERES, Sarajane Marques; BOSCARIOLI, Clodis. **Introdução à Mineração de Dados Com Aplicações em R**. Elsevier Editora Ltda, 2016.

DAVENPORT, T. & PRUSAK, L. **Conhecimento Empresarial: como as organizações gerenciam o seu capital intelectual**, Rio de Janeiro, Campus,1998.

DOMINGOS, Roney. **É #FAKE que vacinas contra a Covid-19 causam a varíola dos macacos**. G1, 2022. Disponível em: https://g1.globo.com/fato-ou-fake/coronavirus/noticia/2022/06/08/e-fake-que-vacinas-contra-a-covid-19-causam-a-variola-dos-macacos.ghtml . Acesso em: 12 out. 2023.

DOMINGOS, Roney. **É #FAKE que variante Ômicron foi inventada para disfarçar efeitos colaterais das vacinas contra a Covid**. G1, 2021. Disponível em: https://g1.globo.com/fato-ou-fake/coronavirus/noticia/2021/12/22/e-fake-que-variante-omicron-foi-inventada-para-disfarcar-efeitos-colaterais-das-vacinas-contra-a-covid.ghtml . Acesso em: 12 out. 2023.

DOMINGOS, Roney. **É #FAKE que vídeo mostre homem chorando sobre corpo de criança morta após tomar vacina contra a Covid-19**. G1, 2022. Disponível em: https://g1.globo.com/fato-ou-fake/coronavirus/noticia/2022/01/12/e-fake-que-video-mostre-homem-chorando-sobre-corpo-de-crianca-morta-apos-tomar-vacina-contra-a-covid-19.ghtml . Acesso em: 12 out. 2023.

Fake news x desinformação: entenda qual é a diferença entre os termos. Tribunal Regional Eleitoral de Goiás**,** 2023. Disponível em: https://www.tre-go.jus.br/comunicacao/noticias/2023/Agosto/fake-news-x-desinformacao-entenda-qual-e-a-diferenca-entre-os-termos . Acesso em: 8 out. 2023.

Fake news. Cambridge Dictionary. Disponível em: https://dictionary.cambridge.org/pt/dicionario/ingles/fake-news . Acesso em: 12 out. 2023.

FÁVERO, Luiz Paulo; BELFIORE, P. **Manual de Análise de Dados**. Elsevier Brasil, 2017.

FAYYAD *et al*. Advances in Knowledge Discovery and Data Mining. Menlo Park: AAAI Press, 1996.
GRUS, J. **Data Science do Zero**. Alta Editora, 2016.

Guerra do Iraque se baseou em "pura ficção", critica ex-analista da CIA. Brasil de Fato, 2016. Disponível em: https://www.brasildefato.com.br/2016/12/02/guerra-do-iraque-se-baseou-em-pura-ficcao-critica-ex-analista-da-cia . Acesso em: 11 out. 2023.

GUIMARÃES, Pedro; RODRIGUES, Cleber. **4 em cada 10 brasileiros afirmam receber fake news diariamente**. CNN Brasil, 2022. Disponível em: https://www.cnnbrasil.com.br/nacional/4-em-cada-10-brasileiros-afirmam-receber-fake-news-diariamente/#:~:text=No%20Brasil%2C%20quatro%20em%20cada . Acesso em: 12 out. 2023.

IBM. Machine Learning Pipeline. Disponível em: https://www.ibm.com/topics/machine-learning-pipeline . Acesso em: 9 mai. 2024.

INDURKHYA, N; DAMERAU, F. J. **Handbook of Natural Language Processing**. Chapman & Hall/CRC, 2ª ed, 2010.

KAMPAKIS, Stylianos. **The Decision Maker's Handbook for Data Science: a Guide for Non Tecnical Executives, Managers and Founders**. London, Apress, 2020.

LO, Frank**; What is Data Science?** DATAJOBS. Disponível em: https://datajobs.com/what-is-data-science . Acesso em: 24 out. 2023.

MARS, A. **Como a desinformação influenciou nas eleições presidenciais?** EL País, 2018 Disponível em: https://brasil.elpais.com/brasil/2018/02/24/internacional/1519484655_450950.html .

MELLO, Daniel**. Fake news são desafios para institutos de estatística, diz presidente do IBGE**. Agência Brasil, 2018. Disponível em: https://agenciabrasil.ebc.com.br/geral/noticia/2018-03/fake-news-sao-desafios-para-institutos-de-estatistica-diz-presidente-do-ibge . Acesso em: 12 out. 2023.

MIGUEL, Paulo Augusto Cauchick. **Metodologia de pesquisa em engenharia de produção e gestão de operações**. 2ª ed. Elsevier Editora Ltda. 2012.

Mortes e casos de coronavírus nos estados | Coronavírus. G1**,** 2023**.** Disponível em: https://especiais.g1.globo.com/bemestar/coronavirus/estados-brasil-mortes-casos-media-movel/ . Acesso em: 12 out. 2023.

MOZILLA DEVELOPER NETWORK. **Element p**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/p . Acesso em: 31 maio 2024.

MOZILLA DEVELOPER NETWORK. **HTML – HyperText Markup Language**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML . Acesso em: 31 mai. 2024.

NETO, Jose Antonio Ribeiro. Big Data para Executivos e Profissionais de Mercado. 2018.

Notícias falsas sobre eleição nos EUA têm mais alcance que notícias reais. G1, 2016. Disponível em: https://g1.globo.com/mundo/eleicoes-nos-

eua/2016/noticia/2016/11/noticias-falsas-sobre-eleicoes-nos-eua-superam-noticias-reais.html . Acesso em: 11 out. 2023.

O que é uma API?. AWS. Disponível em: https://aws.amazon.com/pt/what-is/api/ . Acesso em: 30 out. 2023.

PROVOST, Foster; FAWCETT, Tom. **Data Science para negócios: o que você precisa saber sobre mineração de dados e pensamento analítico de dados**. 1ª ed. Alta Book, 2016.

PYTHON. **Data Structures**. Disponível em: https://docs.python.org/3/tutorial/datastructures.html . Acesso em: 14 maio. 2024.

RAUTENBERG, Sandro; DO CARMO, Paulo Ricardo Viviurka. **Vista do Big data e ciência de dados. Brajis**, 2019. Disponível em: https://revistas.marilia.unesp.br/index.php/bjis/article/view/8315 . Acessado em: 25 out. 2023.

Relembre as mentiras mais famosas de Trump. G1, 2016**.** Disponível em: https://g1.globo.com/mundo/eleicoes-nos-eua/2020/noticia/2020/11/09/relembre-as-mentiras-mais-famosas-de-trump.ghtml . Acesso em: 12 out. 2023.

SANTOS, Virgilio F.M. **O que é análise de dados?** Blog, Seis Sigma. 2016. Disponível em: https://www.fm2s.com.br/analise-de-dados-como-estruturar/ . Acesso em: 27 out. 2023.

SHAH, Chirag. **A hands-on introduction to data science**. Cambridge, United Kingdom. New York, Usa Cambridge University Press, 2020.

SpaCy. **Matcher**. Disponível em: https://spacy.io/api/matcher. Acesso em: 10 mai. 2024.

SpaCy. **Trained Models** & Pipelines. Disponível em: https://spacy.io/models . Acesso em: 9 mai. 2024.

Statistics Canada. **Understanding Basic Statistics**. Disponível em: https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch1/definitions/5214853-eng.htm . Acesso em: 6 abr. 2024.

WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação**. 2ª ed. Elsevier Editora Ltda, 2014.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Gustavo Dias Martins
do Curso de Engenharia de Computação ,matrícula 20181003302069 ,
telefone: 62 982696991 e-mail gustavodiasmartins38@gmail.com , na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado A REALIDADE EM CONSTANTE DEFORMIDADE: CIÊNCIA DE DADOS APLICADA À ANÁLISE DE FAKE NEWS NO BRASIL, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 25 de junho de 2024.

Assinatura do(s) autor(es): gov.br Documento assinado digitalmente
GUSTAVO DIAS MARTINS
Data: 25/06/2024 16:56:19-0300
Verifique em https://validar.iti.gov.br

Nome completo do autor: Gustavo Dias Martins

gov.br Documento assinado digitalmente
ANDRE LUIZ ALVES
Data: 26/06/2024 11:49:55-0300
Verifique em https://validar.iti.gov.br

Assinatura do professor-orientador:

Nome completo do professor-orientador: André Luiz Alves