UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Ronan José Lopes

Construção de uma plataforma para análise de tendências, padrões e interações de usuários em conteúdos publicados no Twitter

São João del-Rei

2022

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Ronan José Lopes

Construção de uma plataforma para análise de tendências, padrões e interações de usuários em conteúdos publicados no Twitter

Dissertação apresentada como requisito para obtenção do título de mestre em Ciências no Curso de Mestrado do Programa de Pós Graduação em Ciência da Computação da UFSJ.

Orientador: Carolina Ribeiro Xavier

Coorientador: Vinícius da Fonseca Vieira

Universidade Federal de São João del-Rei – UFSJ

Mestrado em Ciência da Computação

São João del-Rei

2022

Ronan José Lopes

Construção de uma plataforma para análise de tendências, padrões e interações de usuários em conteúdos publicados no Twitter

Dissertação apresentada como requisito para obtenção do título de mestre em Ciências no Curso de Mestrado do Programa de Pós Graduação em Ciência da Computação da UFSJ.

Trabalho aprovado. São João del-Rei, 17 de novembro de 2022:

Carolina Ribeiro Xavier
Orientadora

Vinícius da Fonseca Vieira
Co-orientador

Edimilson Batista dos Santos
Convidado interno

Michele Amaral Brandão
Convidada externa

São João del-Rei
2022

Agradecimentos

Este trabalho só foi possível graças à reunião de inúmeras causas e condições que o permitissem acontecer. Dentre elas, antes de tudo, a minha existência, desenvolvimento e sustentação só foram possíveis pelo apoio de minha família. A meus pais, minha irmã Elisa, meu irmão Lucas e meus sobrinhos João Lucas, Júlia e Davi, devo uma parte incalculável de quem hoje sou. Devo também o conhecimento adquirido ao longo de minha trajetória aos professores que comigo compartilharam mais que conceitos teóricos e me inspiraram durante minha trajetória acadêmica. Em especial, aos meus orientadores Carol e Vinícius, que sempre me apontaram a direção e tornaram esse percurso menos sinuoso. Em especial também a meu professor Alfredo Aveline (Lama Padma Samten), tutores e companheiros de CEBB, onde encontrei a sensação de pertencimento.

De tantos outros que estiveram sempre ao meu lado, eu falharia em citar nominalmente todos que me apoiaram direta ou indiretamente, mas não poderia deixar de mencionar as relações originadas na UFSJ e cultivadas para muito além dela. Todos os momentos de dificuldade, descontração e parceria forjaram laços que nós, sarcasticamente, chamamos de uma #AmizadeFalsa. Aos meus amigos de adolescência que o J.A.A.C., o corujão e a vida me trouxeram ao longo desses anos, eu só posso agradecer pelos inúmeros momentos memoráveis compartilhados. "Somos amigos em terra, somos amigos no mar. Juntos fomos à guerra, juntos estamos no bar". Tem sido uma jornada e tanto!

Por último, agradeço a todos os erros, acertos e tropeços que me trouxeram até aqui. Todas as lições valiosas advindas desse caminho não seriam possíveis de outra forma que não me permitindo morrer e nascer novamente a cada dia. "Caminhante, não há caminho, se faz o caminho ao andar". A caminhada é a única permanente. Aqui, por enquanto. Em breve, nem isso. Nós, impermanentes seguimos!

Resumo

Há um crescente interesse comum entre as áreas de pesquisa pela análise e descoberta de conhecimento em redes sociais. Para alguns pesquisadores, entretanto, essa tarefa esbarra no empecilho do requisito técnico para implementação (e por consequência, alto custo para terceirização) de uma metodologia para esse tipo de análise. A fim de auxiliar esse processo, desde o agendamento da coleta dos dados, até o pós-processamento e visualização, neste trabalho é apresentada uma plataforma *web* que viabiliza a automatização dessas etapas. A fim de validar a utilização da plataforma desenvolvida, foi realizada (com o auxílio da ferramenta) a análise do evento da leitura do relatório da CPI do COVID no Brasil, ocorrido no mês de Outubro de 2021. Dentre as principais funcionalidades implementadas, destacam-se: modelagem de tópicos, análise de sentimento, nuvem de palavras, usuários mais centrais, potenciais usuários não-autênticos, apresentação de tendências relacionadas no período de coleta e detecção e análise de padrões em comunidades. Os resultados apresentam, além da avaliação de qualidade das funcionalidades implementadas, uma análise qualitativa de informações e conhecimentos que puderam ser extraídos através do uso da própria ferramenta.

Palavras-chaves: ciência de dados, mineração de dados, detecção de comunidades, redes sociais, twitter, política, cpi da covid

Abstract

In nowadays there is a growing interest (common to most of research areas) in data analysis and knowledge discovery in social networks. For some researchers, however, that job stand up against a great technical requirement to put it into practice (and consequently there's also a high cost for outsourcing that job). In order to assist this process all the way from the data mining scheduling to the post-processing and data visualization, this work has the goal to create a web platform to make possible to automate that task. To validate the usability of the platform, a analysis about the CPI COVID report reading in Brazil (which happened in October 2021, at the country senate) was made using the developed tool. Among the features implemented, the main ones are: topic modeling, sentiment analysis, word cloud, most central users, potential non-authentic users, related trends and community detection. The results presented show, beyond the quality analysis of the implemented features, a qualitative analysis of the information and knowledge that could be acquired using the platform.

Key-words: data science, data mining, community detection, social networks, twitter, politics, covid cpi

Lista de ilustrações

Lista de abreviaturas e siglas

API	*Application Programming Interface*
AWS	*Amazon Web Services*
CPI	Comissão Parlamentar de Inquérito
DB	*Database*
CNN	*Convolutional Neural Network*
EC2	*Elastic Compute Cloud*
HTML	*Hyper Text Markup Language*
HTTP	*Hyper Text Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
JSON	*JavaScript Object Notation*
LDA	*Latent Dirichlet Allocation*
ML	*Machine Learning*
MVC	*Model-View-Controller*
NER	*Name Entity Recognition*
NMF	*Negative Matrix Factorization*
REST	*Representational State Transfer*
PLN	Processamento de Linguagem Natural
ŔEST	*REpresentational State Transfer*
RT	*Retweet*
SGDB	Sistema Gerenciador de Banco de Dados
TF-IDF	*Term Frequency — Inverse Document Frequency*
URL	*Uniform Resource Locator*

Sumário

1 Introdução

As redes sociais, como plataformas de criação livre e espontânea de conteúdo por seus usuários, têm se tornado uma fonte importante de pesquisa e aplicação em análise de discursos e tendências, tanto na academia quanto no mercado de trabalho. As informações e *insights* obtidos através dessa investigação permitem, por exemplo, uma definição de estratégia de *marketing* mais assertiva e adaptativa aos eventos em tempo real ao redor mundo (SILVA, 2016). O processamento e análise dos conteúdos publicados permitem sumarizar e ter uma melhor visão geral da repercussão de um tema em particular na rede, bem como dos usuários autores desses conteúdos. Dentre os fatores mais relevantes na tomada decisão, a capacidade de reação aos eventos no menor espaço de tempo possibilita essa melhor adaptabilidade e uma resposta mais ágil.

No Twitter (rede social com o formato de *microblog*, onde as publicações dos usuários são limitadas a um número curto de caracteres) esses conteúdos terminam por se voltar principalmente à emissão de opinião instantânea a respeito dos eventos em foco no momento (chamados *trending topics* na plataforma), local e globalmente. Dadas essas características, a análise de conteúdo na rede é foco de pesquisa como forma de termômetro para a reação imediata dos usuários a diversos eventos (ZHAO, 2011). O interesse na investigação também segue a direção inversa, ou seja, avaliar e entender o papel da rede em relação ao engajamento e formação de opinião dos usuários que consomem seu conteúdo (PARK, 2013). Nesse cenário, algumas adversidades se apresentam, como o desafio em extrair conhecimento de textos curtos, bem como a dificuldade em lidar com particularidades da língua portuguesa em algoritmos e automações.

Alguns outros desafios também surgem ao lidar com esse tipo de informação: devido ao grande volume do fluxo de publicações, encontram-se entraves não somente com relação ao armazenamento e processamento, mas também em extrair informações relevantes desses dados (FRANÇA, 2014). Outro desafio é organizar, delimitar e ter um escopo bem definido a ser trabalhado em relação a todo o *corpus* obtido na coleta das informações. Apesar desses empecilhos, muitos pesquisadores se propõem a explorar o potencial identificado na rede através de soluções computacionais, como o *Hootsuite* (RAPPIE, 2016), por exemplo. Outras soluções disponíveis em formato comercial (como *Unbox Social*[1], *Tweetsmap*[2] e *Tweepi*[3], se apresentam em um formato no qual suas funcionalidades são disponibilizadas sem retrições somente para planos pagos. Dentre as poucas opções abertas (como o *Bot Sentinel*[4], por exemplo), a limitação encontra-se nas funcionalidades

[1] https://www.unboxsocial.com/twitter-analytics-tool
[2] https://tweepsmap.com/
[3] https://tweepi.com/
[4] https://botsentinel.com/

disponibilizadas.

1.1 Justificativa

Apesar de todos os esforços dos pesquisadores e das organizações interessadas em explorar esse potencial de pesquisa, ainda há uma distância muito grande entre as ferramentas disponibilizadas e a demanda dos pesquisadores em utilizar uma plataforma que viabilize uma análise mais aprofundada sem onerar a pesquisa ou exigir um alto conhecimento técnico. Dentre as necessidades, verifica-se a carência de uma plataforma que seja, além de aberta, customizável (uma vez que o pesquisador deve conseguir parametrizar como deseja obter seus dados) e de fácil interação e visualização (para análise efetiva e obtenção de *insights* acerca dos dados obtidos).

Dada a variedade de algoritmos que podem ser utilizados na análise de dados, aqueles que se aplicam ao cenário de interesse de estudo (publicações do Twitter) podem ser implementados e abstraídos de forma a reduzir a complexidade na interação do usuário. Dentre essas abstrações é possível integrar nesse cenário, por exemplo, ferramentas e métodos de redes complexas que permitem entender o comportamento dos indivíduos em torno de determinado assunto (quais são os principais atores, como se agrupam, principais características). Ainda sobre os usuários, em oposição a comportamentos automatizados via robôs, comumente deseja-se saber o nível de organicidade dentro da base de estudo como parâmetro de confiabilidade na autenticidade desses dados.

Com relação à análise do discurso em si, é possível também aplicar técnicas de mineração de texto (PEZZINI, 2017) e processamento de linguagem natural (PLN) que permitam sumarizar e visualizar de forma mais consolidada informações implícitas nos dados coletados. A modelagem de tópicos, por exemplo, permite identificar quais os termos mais importantes e como eles ocorrem simultaneamente (indicando uma relação/associação que talvez não seja trivial em uma inspeção manual). É possível ainda fazer uma análise de sentimento para identificar, temporalmente, qual o teor das opiniões emitidas pelos usuários nas publicações. A integração entre essas e demais funcionalidades dentre as possibilidades presentes pode ser realizada de forma visualmente intuitiva e de fácil usabilidade para o usuário, de forma que o principal objetivo visa reduzir a distância identificada entre os pesquisadores e a ferramenta.

1.2 Objetivos

Dada essa necessidade identificada, o objetivo do trabalho consiste na implementação de uma plataforma *web* que auxilie todo esse processo. Para que o objetivo principal seja alcançado, o trabalho apresenta alguns objetivos específicos: realizar a coleta e ar-

mazenamento de *tweets* através do agendamento, trabalhar o volume de dados coletados e implementar os métodos necessários à obtenção de *insights* pelo usuário da plataforma para, enfim, apresentá-las de forma visualmente intuitiva. Se tratando das funcionalidades presentes na ferramenta, busca-se consolidar em resumos visuais toda a base coletada (com o auxílio de funcionalidades como nuvens de palavras, mapa de calor, gráficos de distribuição, dentre outros), bem como fornecer ainda informações mais específicas e não-triviais através do uso de algoritmos em aprendizado de máquina, redes complexas e ciência de dados em geral. Além da análise de sentimento da publicação e modelagem de tópicos citadas anteriormente, a detecção de comunidades de usuários (bem como a identificação de atores centrais) visa permitir a caracterização dos subgrupos presentes na rede. Essas implementações visam auxiliar a identificação de padrões mais implícitos na base de dados coletada.

Seguindo as boas práticas de engenharia de *software*, é requisito que a ferramenta seja de boa usabilidade, fácil acesso (a escolha de uma plataforma web permite que qualquer dispositivo com navegador possa utilizá-la sem necessidade de instalação) e com um nível de parametrização que permita ao pesquisador flexibilizar e delimitar o escopo da sua análise. Com o objetivo de validar a utilização da plataforma (tanto para investigação de um cenário de interesse e obtenção de *insights* quanto para verificação dos requisitos levantados anteriormente), a ferramenta foi aplicada de forma prática no contexto da leitura do relatório da CPI DO COVID . Através da análise dos resultados obtidos após a coleta, foi possível observar a formação de comunidades de acordo com o posicionamento político, centralizada nos principais atores de cada polaridade, bem como discursos de descontentamento de ambos os lados. Outras descobertas que foram também apoiadas pela ferramenta serão apresentadas posteriormente no capítulo de resultados.

1.3 Organização do Texto

O restante deste trabalho está estruturado da seguinte forma: o Capítulo 2 se dedica a abordar os conceitos necessários para o entendimento das funcionalidades desenvolvidas. No Capítulo 3 são apresentados alguns trabalho relacionados, e no Capítulo 4 são descritos os métodos implementados, bem como detalhes das implementações utilizadas. No Capítulo 5 os resultados obtidos até o momento pela inspeção e análise dessas funcionalidades são apresentados e discutidos. Por fim, o Capítulo 6 apresenta as conclusões do trabalho e propostas de trabalhos futuros.

2 Fundamentação e Referencial Teórico

Para alguns objetivos mapeados na proposta da plataforma, especialmente funcionalidades e algoritmos que se aprofundam em áreas mais específicas da ciência da computação, é necessário ter por base um arcabouço teórico que não é de entendimento trivial. Os conceitos e fundamentos exigidos para melhor compreensão dos métodos utilizados são descritos a seguir.

2.1 *Text Mining*

Atualmente, de acordo com estimativas (TOWARDS AI, 2019), apenas 20% dos dados textuais gerados via mensagens e publicações em mensagens instantâneas e publicações de redes sociais se encontram em formato estruturado (ou seja, que obedeçam um padrão). O formato textual se tratando por si só de um formato altamente não-estruturado, demanda técnicas de processamento e análise para obtenção de informações significativas, principalmente em aplicações que visam automatizar a inspeção de grandes volumes de dados. Nesse contexto, a mineração de texto é o processo de transformar esse conteúdo originalmente não-estruturado em um formato estruturado que permita a identificação de padrões significativos e novos *insights*.

Na área de mineração de texto, ao trabalhar com coleções de documentos (como publicações em *blogs* ou artigos de notícias), uma aplicação comum é dividir essas coleções em grupos naturalmente relacionados para que esses possam ser entendidos separadamente dos demais. A modelagem de tópicos é um método para classificação não-supervisionada de tais documentos. Na prática, a modelagem de tópicos envolve a contagem e agrupamento de padrões de palavras semelhantes para inferir tópicos em dados não estruturados.

Em um exemplo prático de aplicação seria a análise do impacto de um produto após o seu lançamento, para o qual é desejável avaliar a recepção dos usuários. Uma análise manual seria inviável, principalmente quando se trata de uma larga escala de dados coletados. Uma melhor abordagem, para uma visão macro dos principais temas presentes, seria extrair uma modelagem de tópicos da base textual. Através desse recurso, seria possível não só obter os principais termos, mas também a relação e coocorrência entre eles. Dentre as diferentes abordagens que podem ser utilizadas para obtenção desses tópicos modelados, duas foram exploradas na implementação da plataforma: a matriz não-negativa de fatoração (NFM)(LEE; SEUNG, 1999) utilizando a frequência de termos - *Term Frequency — Inverse Document Frequency* (TF-IDF) (STECANELLA, B, 2019) e a *Latent Dirichlet Allocation* (BLEI, 2003), detalhadas nas subseções a seguir.

A frequência de termos - frequência de documentos inversa (TF-IDF) é uma medida estatística que avalia a frequência de uma palavra para um documento em uma coleção de N documentos. O TF-IDF foi criado principalmente para busca de documentos e recuperação de informação. Em sua operação, o algoritmo incrementa o valor de um termo proporcionalmente ao número de vezes que uma palavra aparece em um determinado documento, mas é compensado pelo número de documentos que contêm a palavra. Portanto, palavras comuns em todos os documentos têm uma relevância e classificação baixa, ainda que possam aparecer muitas vezes, já que não significam muito para esse documento em particular. A ocorrência de uma palavra exclusivamente para um determinado documento, por sua vez, vai ter sua importância a depender da razão entre seu número de ocorrências e o total de termos daquele documento específico.

Para detalhar o cálculo do TF-IDF, é necessário definir algumas terminologias:

- t — termo (palavra)
- d — documento (conjunto de palavras)
- N — contagem de elementos no *corpus*
- *corpus* — o conjunto de todos os documentos

Dadas as nomenclaturas, o primeiro índice a ser obtido refere-se à frequência do termo em um documento, que basicamente é o número de vezes que ele ocorre pela quantidade total de palavras no documento:

$$tf(t, d) = count(t)\ em\ d\ /\ count(palavras\ em\ d) \tag{2.1}$$

A frequência do document (df), por sua vez, mede a importância do documento em todo o conjunto do *corpus*, de forma semelhante ao tf. A diferença é que tf é um contador de frequência para um termo t no documento d, enquanto df é a contagem de ocorrências do termo t no conjunto de documentos N. Em outras palavras, df é a razão de documentos em que a palavra está presente. A contagem do documento é considerada se houver pelo menos uma ocorrência, sendo dispensável saber o número total.

$$df(t) = count(documentos\ em\ que\ t\ ocorre)\ /\ count(d) \tag{2.2}$$

Ao calcular tf, todos os termos são considerados igualmente importantes. No entanto, sabe-se que certos termos, considerados *stop words* (como os termos “é”, “de” e “isso”, por exemplo), podem aparecer muitas vezes, mas têm pouca importância semântica. Assim, é necessário ponderar com menores fatores, termos frequentes, enquanto os termos mais raros, escalam com fatores maiores. Isso é feito calculando o idf, um fator de

frequência de documento inverso que mede a informatividade do termo t. O idf pode ser calculado através da razão:

$$idf(t) = N/df \tag{2.3}$$

Ao calcular o idf, o seu valor tende a ser muito baixo para as palavras que mais ocorrem, já que o denominador(df) tende a crescer. Esse cálculo provê uma medida mais interessante, dado seu peso relativo, mas ainda existem alguns outros problemas com o idf: no caso de um corpus grande, o valor do idf pode escalar rapidamente e "explodir". Para evitar esse efeito colateral, utiliza-se o log do idf . Outro ponto a ser considerado durante o cálculo é que, quando ocorre uma determinada palavra que não está no vocabulário, o df (denominador) será 0. Como não é possível dividir por 0, o valor é suavizado adicionando 1 ao denominador. A fórmula do idf, portanto, por fim é dada por:

$$idf(t) = log(N/(df + 1)) \tag{2.4}$$

Multiplicando as duas medidas, obtém-se o valor final para um determinado termo:

$$tf\_idf(t, d) = tf(t, d) * log(N/(df + 1)) \tag{2.5}$$

2.1.1 *NMF - Non-Negative Matrix Factorization*

A fim de responder a pergunta "A percepção do todo é baseada na percepção de suas partes?", os pesquisadores (LEE; SEUNG, 1999) propuseram um algoritmo para fatoração de matriz não-negativa que é capaz de aprender características semânticas de texto, em contraste com outros métodos, que aprendem de forma holística, e não com representações baseadas em partes. Na prática, o NMF é um método estatístico que ajuda a reduzir a dimensão dos *corpus* (documentos de texto) de entrada.

Considerando um caso geral, o modelo teórico pode ser exemplificado com uma matriz de entrada V de dimensões m x n. O método fatora a matriz V em duas matrizes W e H, de modo que a dimensão de W seja m x k e a de H seja n x k. No contexto de aplicação, V representa a matriz de termos por documento: cada linha da matriz H é um *word embedding* e cada coluna da matriz W representa o peso de cada palavra obtida em cada sentença. Uma ilustração prática pode ser visualizada na Figura 1, que demonstra como as matrizes decompostas (W e H) são obtidas de forma que sua multiplicação resulta na matriz original (V).

Em um exemplo de aplicação proposto por (CHOUBEY, V., 2020), suponha um conjunto de dados que consiste em resenhas de filmes de super-heróis. Na matriz de termos do documento (matriz de entrada V), há documentos individuais ao longo das linhas da

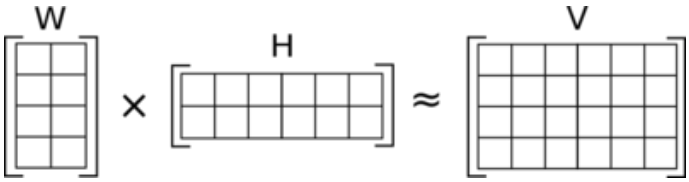


Figura 1 – Ilustração da fatoração das matrizes W, H e V

matriz e cada termo único ao longo das colunas. No caso, a resenha é composta por termos como Tony Stark, *Ironman*, Mark 42 entre outros, que deveriam ser agrupados sob o tópico *Ironman*. Nesse método, cada uma das palavras individuais na matriz de termos do documento é levada em consideração. Durante a fatoração, cada uma das palavras recebe uma ponderação com base na relação semântica entre as palavras. Mas aquele com maior peso é considerado o tópico para um conjunto de palavras. Portanto, esse processo é uma soma ponderada de diferentes palavras presentes nos documentos.

O NMF por padrão produz representações esparsas. Isso significa que a maioria das entradas está próxima de zero e apenas poucos parâmetros têm valores significativos. Isso pode ser usado quando são exigidos menos tópicos. Uma alternativa ao NMF para obtenção de tópicos, o LDA, é explicado na próxima subseção.

2.1.2 *Latent Dirichlet Allocation* (LDA)

O método de modelagem de tópicos por LDA (*Latent Dirichlet Allocation*) proposto por (BLEI, 2003) trata cada documento como um conjunto de termos, e cada tópico a ser mapeado por sua vez, também conterá uma série de palavras associadas. O objetivo do LDA é encontrar tópicos aos quais um documento pertence, baseado no seu conjunto de termos. Em síntese, dada uma estrutura de lista de palavras pré-processada em operações usuais de tratamento de texto, busca-se obter como saída uma estrutura que indique, para cada tópico identificado, a probabilidade de cada termo pertencer a esse tópico.

O conceito pode ser melhor ilustrado através de um exemplo (TERRY-JACK, M., 2019). Suponha uma série de documentos, compostos por N palavras, rotulados genericamente como:

- Doc1: word1, word3, word5, word45, word11, word62, word88 ...
- Doc2: word9, word77, word31, word58, word83, word 2, word49 ...
- Doc3: word44, word18, word52, word36, word64, word11, word20 ...
- Doc4: word85, word62, word19, word4, word30, word94, word67 ...
- Doc5: word19, word53, word74, word79, word45, word39, word54 ...

A lista acima representa 5 documentos contendo à sua frente a lista de suas palavras, ordenadas por frequência de ocorrência. O objetivo é inferir as palavras de cada tópico, como mostra um exemplo de saída na figura 2. Cada linha na tabela representa um tópico diferente e cada termo está representado na coluna. O valor correspondente indica a probabilidade de que uma determinada palavra pertença ao tópico.

	Word1	word2	word3	word4	.....
Topic1	0.01	0.23	0.19	0.03	
Topic2	0.21	0.07	0.48	0.02	
Topic3	0.53	0.01	0.17	0.04	

Figura 2 – Exemplo de mapeamento da probabilidade entre os termos e tópicos presentes em uma base de texto

Para encontrar os termos significativos para um determinado tópico:

- Uma possibilidade é classificar as palavras em relação à sua pontuação de probabilidade. As primeiras x palavras são escolhidas para representar o tópico. Se x=10, por exemplo, as 10 primeiras palavras são selecionadas. Esta etapa pode não ser necessária, caso o *corpus* (conjunto de termos da base) seja pequeno.
- Como alternativa, é possível definir um limiar na pontuação. Todas as palavras em um tópico com pontuação acima desse limiar podem ser armazenadas como seu representante, ordenadas pela pontuação.

Para entender o funcionamento do LDA, é necessário evidenciar algumas premissas:

- Cada documento é apenas uma coleção de palavras (ou um “saco de palavras”). Assim, a ordem das palavras e o papel gramatical das palavras (sujeito, objeto, verbo, etc) não são considerados no modelo.
- *Stopwords* não carregam nenhuma informação sobre os tópicos e, portanto, podem ser eliminadas dos documentos como uma etapa de pré-processamento. Na verdade, é possível eliminar palavras que ocorrem em pelo menos %80 %90 dos documentos, sem perder nenhuma informação. Por exemplo: se o *corpus* contiver apenas documentos médicos, palavras como "humano", "corpo", "saúde", dentre outras podem estar presentes na maioria dos documentos e, portanto, podem ser removidas, pois não adicionam nenhuma informação específica que faça o documento ser mais relevante.
- Sabe-se de antemão a quantidade de tópicos a serem encontrados. Esse número, dado pela constante k, é definido previamente pelo programador.

Em termos de funcionamento, o algoritmo do LDA pode ser considerado em 2 partes:

- Percorra cada documento e atribua aleatoriamente cada palavra no documento a um dos k tópicos (k é escolhido de antemão).
- Para cada documento d, percorra cada palavra w e calcule:
    1. **p(tópico t | documento d)**: a proporção de palavras no documento d que são atribuídas ao tópico t. Essa probabilidade tenta identificar quantas palavras pertencem ao tópico t para um determinado documento d (excluindo a palavra atual). Se muitas palavras de d pertencem a t, é mais provável que a palavra w pertença a t.
    2. **p(palavra w| tópico t)**: a proporção de atribuições ao tópico t sobre todos os documentos que têm esta palavra w. Tenta capturar quantos documentos estão no tópico t por causa da palavra w. O LDA representa documentos como uma mescla de tópicos. Da mesma forma, um tópico é uma mescla de palavras. Se uma palavra tem alta probabilidade de estar em um tópico, todos os documentos com a palavra w estarão mais fortemente associados a t também. Da mesma forma, se w não for muito provável de estar em t, os documentos que contêm w terão uma probabilidade muito baixa de estar em t, porque o restante das palavras em d pertencerá a algum outro tópico e, portanto, d terá uma maior probabilidade para esses tópicos. Portanto, mesmo que w seja adicionado a t, não trará muitos desses documentos para t.
- Atualize a probabilidade da palavra w pertencer ao tópico t:

$$p(palavra\ w\ com\ tópic\varnothing t) \ = \ p(tópico\ t \mid documento\ d) \ * \ p(palavra\ w \mid tópico\ t) \tag{2.6}$$

A fim de exemplificar uma aplicação, suponha um cenário com 2 tópicos que podem ser classificados como *CAT_related* e *DOG_related*. Cada tópico tem probabilidades associadas para cada palavra, então termos como "leite", "miado" e "gatinho" terão uma probabilidade maior no tópico *CAT_related* do que no *DOG_related*. Este por sua vez, da mesma forma, terá altas probabilidades para palavras como "filhote", "latido" e "osso". Ao considerar um documento contendo as seguintes frases:

- "Cães gostam de mastigar ossos e pegar gravetos".
- "Os cachorros bebem leite."
- "Ambos gostam de latir."

É trivial inferir que o documento pertence ao tópico *DOG_related* porque contém palavras como “Cães”, “ossos”, “cachorros” e “latir”. Apesar de conter a palavra “leite” que pertence ao tópico *CAT_related*, o documento pertence a *DOG_related* pois mais palavras correspondentes. Para um conjunto mais amplo e complexo de termos, entretanto, essa conclusão pode não ser tão trivial, especialmente para a língua portuguesa. Algumas dificuldades são conhecidas nos algoritmos de classificação, como por exemplo, trabalhar com palavras homógrafas do idioma (HAMADA; NETO, 2015), ou seja: palavras que possuem a mesma grafia mas possuem significados diferentes e que podem induzir ao erro.

2.2 Grafos, Redes Complexas e Aplicações

Na ciência da computação, um grafo é um tipo de dados abstrato que deriva da estrutura matemática do modelo (WEST, 2001), usada para modelar relações de pares entre objetos. Um grafo neste contexto é composto de vértices que são conectados por arestas. É possível definir um grafo como não-direcionado (onde as arestas ligam dois vértices simetricamente) ou direcionado (nos qual a direção da ligação é relevante). Essa ligação pode ainda possuir um valor numérico correspondente, que pondera a aresta e consequentemente torna o grafo ponderado. Na definição formal, o grafo é um par ordenado G=(V,E), tal que:

- V é um conjunto de vértices (também chamados nós ou pontos);
- $E \subseteq \{\{x, y\} \mid x, y \in V \text{ e } x \neq y\}$ é o conjunto de arestas (também chamadas de *links* ou linhas), que consistem de uma lista não-ordenada de pares de vértices (indicando uma ligação entre os mesmos)

No contexto da teoria de redes, uma rede complexa é um grafo (rede) com características topológicas não triviais – características que não ocorrem em redes simples, como grafos aleatórios, mas geralmente ocorrem em redes que representam sistemas reais. O estudo de redes complexas é uma área recente e ativa de pesquisa científica (ALBERT; BARABÁSI, 2002) inspirada em grande parte por descobertas empíricas de redes do mundo real, como redes de computadores, redes biológicas, redes tecnológicas, redes cerebrais, redes climáticas e redes sociais.

Em redes complexas, cada nó possui algumas características únicas que definem a importância do nó com base no contexto específico da aplicação (SAXENA; IYENGAR, 2020). Essas características podem ser identificadas por meio de diversas métricas de centralidade definidas na literatura. Algumas dessas medidas de centralidade podem ser calculadas usando informações locais do nó, como a centralidade de grau de entrada.

Outros métodos usam informações globais da rede como centralidade de proximidade (*closeness*), centralidade de intermediação (*betweenness)*, centralidade de autovetor (*eigenvector)*, PageRank, dentre outros.

As subseções seguintes abordam, para aquelas medidas de centralidade utilizadas no desenvolvimento da plataforma, seus conceitos e principais características. Além disso, são explorados outros dois métodos/aplicações em redes complexas utilizadas na implementação de funcionalidades da plataforma: o *backbone* de Serrano (SERRANO M. BOGUÑÁ, 2009), que consiste em uma sub-rede que representa a "espinha dorsal"(ou seja, os nós mais importantes na topologia da rede) e a detecção de comunidades de Louvain (BLONDEL V. GUILLAUME, 2008), utilizada para identificação de *clusters* ou grupos de nós em uma rede.

2.2.1 Grau de Entrada

A medida de centralidade mais intuitiva consiste em selecionar o nó que possui o maior número absoluto de arestas incidentes nele. Embora seja a mais trivial e reflita um aspecto de interesse, sua análise deve ser cautelosa já que a métrica por si só pode não representar corretamente a realidade em alguns casos. Se tratando de redes sociais, por exemplo, é conhecido que o número absoluto de seguidores não necessariamente reflete em um engajamento orgânico, o qual geralmente é o de interesse nesse contexto. Ainda assim, é uma métrica importante para verificar quais são os usuários de maior destaque.

A Figura 3 ilustra um exemplo do nó mais importante em uma rede segundo essa métrica. É possível verificar através da imagem que os vértices com coloração mais quente são aqueles que possuem o maior número absoluto de conexões com outros nós. Essa representação visual (mapa de calor) auxilia para redes com um número de nós que inviabilize essa identificação imediata, de forma que os vértices com menor relevância (menor grau de entrada) recebem a cor fria e os mais centrais (maior grau de entrada) ficam em evidência na representação visual.

2.2.2 *Betweenness* (intermediação)

A centralidade *Betweenness* (ou "intermediação"), por sua vez, é uma medida baseada em caminhos mais curtos, seguindo a definição formal elaborada por (FREEMAN, 1977). Para cada par de nós na rede, existe pelo menos um caminho mais curto entre os vértices (desde que eles sejam atingíveis um pelo outro) tal que o número de arestas pelas quais o caminho passa é minimizado (já que foram consideradas arestas não-ponderadas, caso contrário seriam trabalhadas as somas dos pesos). A centralidade de *Betweenness* para determinado nó é o número desses caminhos mais curtos que passam por ele.

Na prática, a centralidade de intermediação é uma forma de detectar a importân-

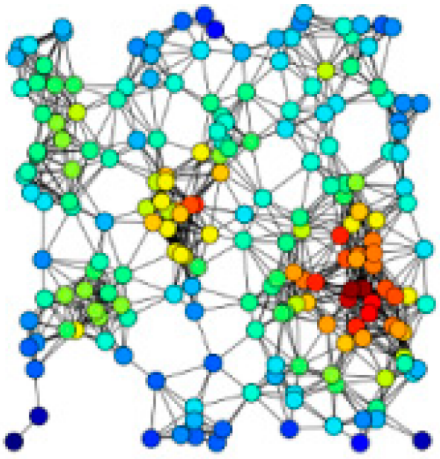


Figura 3 – Exemplo de nós com maior grau de entrada (mapa de calor)

cia que um nó tem na passagem do fluxo de informações em um grafo. É frequentemente usado para encontrar nós que servem como uma ”ponte“ de uma parte de um grafo para outra. Essa medida tem aplicações em uma ampla gama de problemas na teoria de redes, incluindo problemas relacionados a redes sociais, biologia, transporte e cooperação científica. Na análise de redes sociais, por exemplo, a centralidade de intermediação pode ter diferentes implicações (KOURTELLIS, 2013). De uma perspectiva macroscópica, as pontes ou "buracos estruturais"(indicados pela alta centralidade de intermediação) implicam um alto poder de influência, porque permitem que a pessoa na posição exerça o controle do fluxo sobre as pessoas com as quais se conecta (por exemplo, decidir compartilhar informações ou não). No entanto, do ponto de vista microscópico, das redes-ego (ou seja, considerando apenas conexões de primeiro grau), uma alta centralidade de intermediação é correlacionada com indicações de amigos mais próximos (ou seja, fortes laços interpessoais).

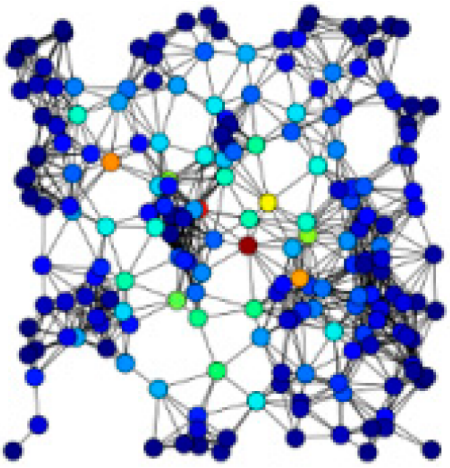


Figura 4 – Mapa de calor utilizando a métrica de intermediação (*betweenness*)

A Figura 4 exemplifica um caso de nó mais central utilizando o *betweenness*. Em

oposição à figura anterior, é possível verificar diretamente um mapa mais frio, com poucos vértices que se sobressaem aos demais. Conforme explicado anteriormente, a importância desses nós se dá pela concentração do fluxo da rede que passa por eles. Por exemplo, em uma rede de telecomunicações, um nó com maior centralidade de intermediação teria mais controle sobre a rede, pois mais informações passarão por esse nó e poderia, portanto, ser um considerado um nó mais crítico em caso de uma eventual indisponibilidade.

2.2.3 Page Rank

Popularmente identificada pela criação dos fundadores do Google (Larry Page e Sergei Brin) mas formalmente elaborado no trabalho desenvlvido em (PAGE, 1999), o algoritmo e sua aplicação em centralidade de PageRank é uma variante da centralidade por auto-vetor *EigenCentrality*, projetada para classificar o conteúdo da web, usando *hiperlinks* entre as páginas como medida de importância. Entretanto, ele pode ser aplicado para qualquer tipo de rede que mapeie interações entre os seus elementos.

A principal diferença do PageRank em relação ao auto-vetor é que ele leva em consideração a direção da aresta/relação. Cada nó em uma rede recebe uma pontuação com base em seu número de arestas de entrada. Esses links também são ponderados considerando a pontuação relativa de seu nó de origem. A implicação prática é que os nós com alto grau de entrada são influentes e os nós aos quais estão conectados compartilham um pouco dessa influência. Assim como a centralidade por auto-vetor, o PageRank pode ajudar a descobrir nós influentes ou importantes cujo alcance se estende além de suas conexões diretas. Sua aplicação é especialmente útil em cenários nos quais a direção da aresta é importante, como por exemplo:

- Redes de colaboração acadêmica: assim como o princípio utilizado no *rankeamento* de páginas web, ao se avaliar uma rede de colaboração científica, não somente o número absoluto de referências, mas também a relevância de quem referencia (ou seja, o autor que cita uma publicação) é importante.
- O próprio Twitter utiliza uma versão personalizada do PageRank (GUPTA, 2013) para recomendar a seus usuários quem eles gostariam de seguir ("*who to follow*").
- Em geoprocessamento, o PageRank tem sido utilizado para classificar espaços (locais) e ruas para prever quantas pessoas (pedestres ou veículos) se deslocalm até eles(JIANG, 2009).

A figura 5 exemplifica bem essa importância "colaborativa"na qual a medida de influência de um nó reflete nos nós mais próximos. Examinando a mesma figura dos exemplos anteriores, é possível verificar que o vértice vermelho que se destacou isoladamente

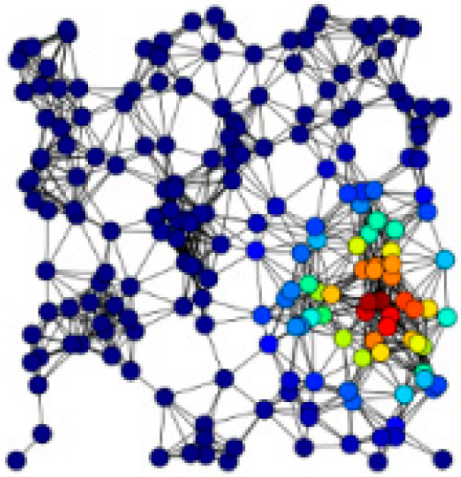


Figura 5 – Mapa de calor utilizando a métrica de PageRank

pelo critério de grau entrada, aqui é classificado como um vértice frio (de baixa relevância), enquanto os nós mais quentes se concentram em uma pequena região.

2.2.4 *Backbone* de uma rede - Algoritmo de Serrano et al.

Além da medida de importância dos vértices consideradas na seção anterior, outro ponto de interesse no estudo de redes complexas é verificar a mesma medida para as arestas do grafo. Principalmente em dependência do fenômeno a ser verificado na rede, nem todas as arestas são igualmente importantes e eventualmente podem representar interações esporádicas e com menor importância. De forma mais específica com o contexto do trabalho, em uma rede que representa a interação de usuários no Twitter, nem todas as interações possuem o mesmo peso e muitas delas podem ser ruidosas. Em contrapartida, algumas dessas interações podem persistir e indicar uma interação mais orgânica e relevante na definição de padrões e relações na rede observada.

Identificar essas relações pode não ser uma tarefa trivial, a considerar as dimensões da rede e a natural heterogeneidade que ocorre nas interações orgânicas. Entretanto, a relevância das arestas (interações) pode ser definida utilizando diferentes suposições acerca de aspectos específicos. Muitos trabalhos na literatura se propuseram a explorar métodos para filtrar as arestas mais fortes/importantes (BRÖHL; LEHNERTZ, 2019). Uma das estratégias mais triviais seria simplesmente remover as arestas menos pesadas de acordo com a definição de um limiar de corte (*threshold*). Essa estratégia simples, ainda que amplamente utilizada, segue uma abordagem ingênua, uma vez que não considera sobre a distribuição dos graus dos vértices na rede.

Uma opção mais elaborada e utilizada na implementação da plataforma foi o filtro de disparidade (SERRANO M. BOGUÑÁ, 2009) para extração do *backbone* da rede. O *backbone* de um grafo ou de uma rede complexa consiste em um subconjunto dos vértices

originais que melhor carregam as características da rede, reduzindo a complexidade em termos de escala. O filtro de disparidade pode reduzir a rede de forma significativa sem destruir a natureza topológica da rede. Esse método funciona como um teste de significância estatística para as arestas ponderadas de uma rede: dada uma distribuição de pesos de arestas, o algoritmo identifica aqueles que formam o *backbone* da rede, usando o equivalente a um limiar de significância estatística para selecioná-los.

Para cada aresta (v_i, v_j) no conjunto E para a rede G, o filtro de disparidade calcula se a probabilidade de sua existência α_{ij} é consistente com seu peso w_{ij}, que diz respeito a uma significância estatística α. As arestas com $\alpha_{ij} < \alpha$, ou seja, arestas que rejeitam a hipótese nula, são consideradas como flutuações anômalas do modelo nulo e devem ser retidas no *backbone*. Em nosso contexto, essas bordas salientes são consideradas interações anômalas entre usuários, que são retidas no *backbone*.

A função do filtro de disparidade, que determina a relevância de uma aresta, consiste em realizar três operações:

1. Obter a distribuição de grau, grau de entrada ou grau de saída do grafo, assim como a lista de arestas da rede. Para grafos não-direcionadas, simetrizar a lista para que o algoritmo seja executado duas vezes em cada aresta.

2. Para cada nó do grafo de grau (de entrada ou saída) superior a 1, calcular a soma dos pesos das arestas de sua rede-ego (ou seja, os vértices que se conectam diretamente a ele).

3. Para cada aresta presente nessa rede, teste seu peso em relação a um modelo nulo (descrito em (MASLOV; SNEPPEN, 2002)) e armazene-a caso passe no teste.

O motivo para executar o filtro duas vezes em cada aresta de uma rede não-direcionada é a necessidade de uma execução para testar a aresta entre os nós i e j usando a soma dos pesos das arestas na rede-ego do nó i, enquanto a outra execução testa a mesma aresta usando a soma dos pesos na rede-ego do nó j. Com o auxílio de alguma biblioteca para manipulação de grafos, tornam-se triviais todas as operações de rede mencionadas acima. O que necessita ser implementado são as iterações para cobrir os passos (2) e (3), juntamente com o código para testar as arestas (ou seja, para gerar seu valor *alpha*).

Esse valor *alpha* é determinante para definir se, dado um valor parametrizável de corte, determinada aresta permanece ou não no grafo resultante (ou o *backbone* da rede). Ao definir esse limiar α, é possível controlar o nível de significância das arestas que são extraídas da rede entre uma estratégia mais agressiva ou uma mais flexível (com valores menores e maiores de α, respectivamente). O nível de flexibilidade ou rigidez desse limiar de corte dependerá, portanto, do contexto a ser aplicado.

2.3 Detecção de Comunidades em Redes Complexas

No estudo de redes complexas, uma das características mais relevantes que representam sistemas reais é a estrutura de comunidades, ou seja, a organização dos vértices em *clusters*, em que muitas arestas unem vértices de um mesmo *cluster*, mas comparativamente há poucas arestas unindo vértices de diferentes *clusters*. Tais agrupamentos, ou comunidades, podem ser considerados como elementos mais fortemente relacionados em um grafo, desempenhando um papel semelhante de acordo com o contexto da modelagem. Detectar comunidades é de grande importância em sociologia, biologia e ciência da computação em geral (LU, 2018). A detecção de comunidades em redes complexas tem despertado grande interesse em pesquisa e desenvolvimento, uma vez que sua aplicação pode encontrar algumas informações úteis escondidas nas redes.

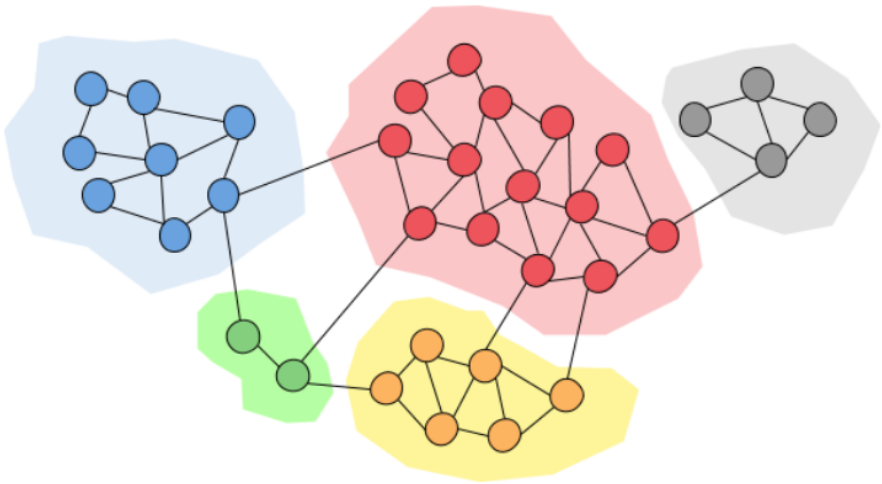


Figura 6 – Exemplo de separação de comunidades em uma rede simples <Disponível em https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae>

A Figura 6 exemplifica, para um grafo aleatório qualquer, uma possível subdivisão de comunidades, delimitadas na visualização pelo uso de cores distintas. É importante notar que, para cada comunidade, os nós presentes possuem um elevado número de interações entre si, ao passo que poucas arestas incidem para vértices de outras comunidades. Essa característica indica uma alta similaridade entre os vértices de uma comunidade ao mesmo tempo que se mostram dissimilares aos vértices das outras comunidades.

2.3.1 O Algoritmo de Louvain

O algoritmo Louvain (BLONDEL V. GUILLAUME, 2008) para detecção de comunidades é um método utilizado para extrair comunidades de grandes redes. Na prática, é um processo de otimização guloso que pode ser executado no tempo $O(n\ log\ n)$, onde n é o número de nós da rede. Esse método tem o objetivo de maximizar a modularidade à medida que o algoritmo avança. A modularidade é um valor de escala entre -0,5 (agru-

pamento não modular) e 1 (agrupamento totalmente modular) que mede a densidade relativa das bordas dentro das comunidades em relação às bordas fora das comunidades. A otimização desse valor teoricamente resulta no melhor agrupamento possível dos nós de uma determinada rede. Mas como iterar por todas as possibilidades dos nós em grupos é inviável em tempo computacional, algoritmos heurísticos são usados.

No método aglomerativo de detecção de comunidades de Louvain, primeiro pequenas comunidades são encontradas otimizando a modularidade local dos nós. Então cada pequena comunidade é agrupada em um outro nó e o primeiro passo é repetido. O método é semelhante ao método proposto anteriormente por (CLAUSET, 2004) que conecta comunidades visando também a otimização de modularidade. Esse valor, a ser otimizado, é definido como um valor no intervalo [-1/2,1], que mede a densidade de *links* dentro de comunidades em relação aos links entre comunidades. Para um gráfico ponderado, a modularidade é definida como:

$$Q = \frac{1}{2}\sum_{ij}\left[A_{\mathrm{ij}} - \frac{kikj}{2m}\right]\delta(ci, cj) \tag{2.7}$$

- A_{ij} representa a aresta entre os nós i e j
- k_i e k_j são a soma dos pesos das arestas dos nós i e j, respectivamente
- m é a soma do peso de todas as arestas no grafo
- c_i e c_j são comunidades dos nós e
- δ é função delta de Kronecker (δ(x,y)=1 se x=y, 0 caso contrário)

Para maximizar esse valor de forma eficiente, o Método de Louvain possui duas fases que se repetem iterativamente. Primeiro, cada nó da rede é atribuído à sua própria comunidade. Então, para cada nó i, a mudança na modularidade é calculada para a remoção de i de sua própria comunidade, movendo-o para a comunidade de cada vizinho j. Esse valor é calculado por duas etapas:

1. Remova i de sua comunidade original
2. Insira i na comunidade de j. As duas equações são bastante semelhantes, e a equação para o passo (2) é:

$$\Delta Q = \left[\frac{\sum_{\mathrm{in}} + 2k_{\mathrm{i,in}}}{2m} - \left(\frac{\sum_{\mathrm{tot}} + ki}{2m}\right)^2\right] - \left[\frac{\sum_{\mathrm{in}}}{2m} - \left(\frac{\sum_{\mathrm{tot}}}{2m}\right)^2 - \left(\frac{ki}{2m}\right)^2\right] \tag{2.8}$$

Onde Σ_{in} é a soma de todos os pesos das arestas dentro da comunidade para a qual i está se movendo, Σ_{tot} é a soma de todos os pesos das arestas que apontam para algum nó na comunidade em que i está se movendo, k_i é o grau ponderado de i, $k_{i,in}$ é a soma dos pesos das arestas entre i e outros nós na comunidade para a qual i está se movendo, e m é a soma dos pesos de todas as arestas na rede. Então, uma vez que este valor é calculado para todas as comunidades às quais i está conectado, i é colocado na comunidade que resultou no maior aumento da modularidade. Se nenhum aumento for possível, i permanece em sua comunidade original. Este processo é executado iterativamente e sequencialmente a todos os nós até que nenhum aumento de modularidade possa ocorrer. Uma vez atingido este máximo local de modularidade, a primeira fase terminou.

Na segunda fase do algoritmo, todos os nós de uma mesma comunidade são agrupados e é construída uma nova rede onde os nós são as comunidades da fase anterior. Quaisquer arestas entre nós da mesma comunidade agora são representados por *self-loops* no novo nó da rede e arestas de múltiplos nós em uma mesma comunidade para um mesmo nó em uma comunidade diferente são representados por arestas ponderadas entre comunidades. Uma vez que a nova rede é criada, a segunda fase terminou e a primeira fase pode ser reaplicada à nova rede.

2.4 Identificação de Usuários não-autênticos

Diversas plataformas de redes sociais (incluindo o Twitter[1]) disponibilizam, através de uma integração via API, a criação de aplicações que possibilitam a automatização de ações dentro da plataforma. Essa facilidade abre uma ampla gama de possibilidades de interações úteis para os usuários da rede. O perfil de usuário @remindme_ofthis na rede, por exemplo, funciona como um robô da seguinte forma: quando qualquer usuário da plataforma deseja ser lembrado de revisitar determinado conteúdo em algum tempo, só precisa comentar na publicação mencionando o robô, seguido do tempo em que o lembrete deve ocorrer ("@remindme_ofthis 3 days", por exemplo).

Entretanto, essa facilidade abre margem também para usos de cunho negativo, como *spam*, importúnio e mais recentemente foram verificados também casos de usuários automatizados que tentam simular comportamento usuários autênticos para disseminação de conteúdos falsos (*fake news*) (SHAO, 2017). Especialmente no contexto político, esse tema tem estado em voga, inclusive sendo estudo de como a rede pode ter influenciado, por exemplo, as eleições presidenciais dos EUA em 2016, nas quais o candidato Donald Trump foi eleito (BOVET; MAKSE, 2019).

Dados esses recentes acontecimentos, há um extenso esforço por parte dos pesquisadores (ainda que reconhecidas todas as dificuldades conforme mostra (KNIGHT, W.,

[1] https://developer.twitter.com/en

2022)) em busca de uma eficiente identificação e intervenção em relação a usuários não-autênticos na rede, que não remetem a um perfil pessoal mas à tentativa de simulação de um perfil orgânico. Uma vez que esse tipo de usuário e os conteúdos disseminados por ele comprometem a confiabilidade da informação, é de interesse geral avançar nos estudos desses algoritmos. Dentre as possibilidades na literatura, a implementação realizada na plataforma de um algoritmo não-supervisionado foi motivada pela proposta em BigDataProject3[2] para detecção automática de perfis falsos.

Esse conjunto de diretrizes propõem um método de avaliação automática para cada usuário, baseado em verificações pontuais que incrementam ou decrementam um *score* desse perfil de acordo com características que indicam a autenticidade de um perfil em rede social. O trabalho desenvolvido por (CRESCI, 2015), por exemplo, ilustra uma aplicação desse conceito que os resultado foi um classificador generalista o suficiente para evitar o *overfitting*, de rápida execução graças ao uso das características menos onerosas e com resultados bem satisfatórios. As regras originais das pesquisas consistem em:

1. O usuário possui nome;
2. O usuário possui imagem de perfil;
3. O usuário possui endereço cadastrado;
4. O usuário possui informação de biografia;
5. A conta tem pelo menos 30 seguidores;
6. A conta foi listada por outro usuário do Twitter
7. Foram escritos pelo menos 50 *tweets*;
8. A conta tem geolocalização habilitada;
9. O perfil contém uma URL;
10. O usuário foi incluído nos favoritos de algum outro usuário;
11. Os tweets escritos possuem pontuação;
12. Foi utilizada uma *hashtag* (#) em pelo menos um *tweet*
13. Utilizou o Twitter por um dispositivo iPhone;
14. Utilizou o Twitter um dispositivo Android;
15. Está conectado ao Foursquare;
16. Está conectado ao Instagram;
17. Logou pelo *site* do Twitter (*twitter.com*);
18. Tweetou o ID de outro usuário, seja através de *retweet* ou menção direta (@);
19. Publica conteúdos que não são somente URL;
20. 2*número de seguidores > número de usuário; de usuários seguindo;
21. Pelo menos um dos *tweets* foi *retweetado* por outra conta; *L*ogou no Twitter por navegadores clientes diferentes

[2] https://github.com/lnyaki/BigDataProject3

Entretanto, algumas das diretrizes não podem ser verificadas na base da aplicação aqui desenvolvida, pois se referem a características mais complexas que envolvem a análise de publicações do usuário (minerar essas informações para uma base tão grande seria inviável devido à limitação de requisições da API). Portanto, na avaliação implementada (a ser descrita na metodologia), foi utilizado um conjunto composto parcialmente dessas diretrizes com a adição de outras verificações viáveis dentro do conjunto de dados coletados.

2.5 Processamento de Linguagem Natural e Análise de Sentimento

No contexto de análise de conteúdos textuais, é crescente o interesse por métodos que possam extrair e sumarizar informações de forma automática, dado o potencial volume de dados coletados. Naturalmente, muitas dessas técnicas são de interesse de aplicação no estudo em redes sociais, a fim de se entender melhor algumas conotações dos conteúdos publicados pelos usuários. Os autores em (OLIVEIRA, 2020), por exemplo, fizeram uso dessas técnicas para identificação de notícias falsas e tendências em redes sociais. Especialmente nesse cenário de aplicação, um desses métodos que pode agregar mais informações à base é o de análise de sentimento, uma vez que as redes sociais são pontos focais de publicação de opiniões de diversos produtos e assuntos.

Na aplicação desenvolvida, optou-se por investigar os resultados através de duas formas de análise de sentimento: a primeira, mais trivial e utilizadas em aplicações semelhantes (NASCIMENTO, 2014), é feita através do uso de dicionários contendo pontuações relativas às palavras do idioma. A segunda, por sua vez, utiliza-se de uma base pré-classificadas de *tweets* no idioma de interesse para, utilizando um algoritmo de redes neurais, treinar e criar um classificador que rotule os conteúdos coletados. Esta seção destina-se a evidenciar e detalhar como foram desenvolvidos esses métodos na aplicação.

2.5.1 Pré-Processamento em Linguagem Natural

Tradicionalmente, os métodos de processamento em linguagem natural realizam algumas etapas anteriores (pré-processamento) para evitar e corrigir alguns empecilhos ao se tratar esse tipo de dado. O objetivo do pré-processamento é transformar essas informações para um formato com melhor normalização e consequentemente, melhorar seu desempenho nos métodos que fazem uso das mesmas, uma vez que esses algoritmos não possuem a capacidade humana de leitura e interpretação. As etapas aplicadas no contexto da aplicação consistem em:

- **Normalização dos caracteres para letras minúsculas (*lower case*)**: Um dos empecilhos que podem surgir na comparação de strings e identificação de padrões

no geral é a variação de um mesmo termo em relação à caixa alta/caixa baixa: seus caracteres podem se apresentar como maiúsculos ou minúsculos. A fim de se trabalhar na mesma norma, todos os caracteres são transformados para *lower case* (minúsculos).

- **Remoção de caracteres especiais e URL's**: No contexto de processamento de linguagem natural e, mais especificamente análise de sentimentos, alguns termos presentes no texto não agregam nenhuma informação/conotação ao conteúdo, e portanto são removidos na etapa de pré-processamento, como no caso de símbolos especiais (como por exemplo $, *, dentre outros) e *links*/URL's que apontam para alguma fonte externa. Uma ressalva nessa exclusão são caracteres correspondentes a *emojis* que representam emoção de alegria ou tristeza (tais como ":)"e ":("), pois estes são relevantes à análise de sentimento. Nesse caso, esses caracteres são substituídos pelos *tokens* "emocaopositiva"e "emocaonegativa".

- ***Lemmatization* (ou Lemitização)**: Outra normalização importante ao se trabalhar com processamento de linguagem natural é identificar e reduzir variações de um mesmo termo. Um verbo, por exemplo, pode estar conjugado em diferentes tempos, mas conter um mesmo significado. Uma opção seria utilizar a técnica de *Stemmatization* (ou Stemização), que reduz a palavra ao seu radical. Entretanto, isso pode produzir alguns resultados inesperados (como as palavras "carreira"e "carro"sendo reduzidas ao radical "carr"). Outra opção, aqui utilizada, foi a Lemitização, que reduz os termos e suas variações a um *lemma*, que se refere a uma palavra que realmente existe na gramática e é mais coerente na aplicação de análise de sentimento.

- **Remoção de *Stopwords***: A fim de se analisar dados textuais e construir modelos mais assertivos, também são removidas da publicação palavras sem grande relevância de significado e que possam não agregar muito valor ao modelo. Geralmente, essas palavras se referem às classes gramaticais de artigo, preposição, interjeição, dentre outras (como por exemplo os termos "a","de","para","com". A remoção dessas palavras é feita utilizando um dicionário que contenha a lista das *stopwords* do idioma em questão.

2.5.2 Análise de Sentimento utilizando dicionário de sentimentos

A primeira forma de analisar a polaridade (positiva, negativa ou neutra) de um conteúdo publicado consiste de um método mais trivial em realizar esse cálculo com base em um dicionário que já pré-determina a polaridade de palavras presentes no idioma. Após carregar esse dicionário para um objeto que permita a consulta termo a termo, que serão percorridos em cada publicação. Assim, o *score* do sentimento de uma publicação é dada pela soma do *score* de seus termos (considerando que, aqueles termos não estiverem

```
    -> seja 'frase' uma string com o conjunto de termos
      pré-processados:
    l_sentimento = []
    for p in frase:
        l_sentimento.append(dic_palavra_polaridade.get(p))
    score = sum(l_sentimento)
    if score > 0:
        return 'Positivo: {} '.format(score)
    elif score == 0:
        return 'Neutro: {} '.format(score)
    else:
        return 'Negativo: {}'.format(score)
```

Listing 1: Pseudocódigo para análise de sentimentos utilizando dicionário

presentes no dicionário de polaridade, tais como os termos neutros, não agregam valor ao *score* da publicação).

O pseudocódigo 1 explicita melhor o conjunto de instruções para obtenção dos valores. Embora de trivial entendimento e implementação, essa abordagem conta com alguns empecilhos, por exemplo lidar com quantificações (termos como "muito", "bastante","imensamente"devem ser considerados na análise) e outras particularidades da linguagem natural, como por exemplo o uso de ironia e sarcasmo. Entretanto, ainda que considerada uma abordagem "ingênua", ela pode ser validada como ponto de partida e de comparação para *double check* com o método por aprendizado de máquina, descrito a seguir.

2.5.3 Análise de Sentimento utilizando Aprendizado de Máquina (Redes Neurais)

Para uma abordagem mais profunda e que pudesse utilizar dados já conhecidos, optou-se por utilizar um modelo da biblioteca spaCy[3] que possui um treinamento com base em *tweets* já classificados para criação de um classificador a ser utilizado na base coletada. Enquanto os detalhes técnicos da implementação são detalhados na próxima seção (materiais e métodos), esta destina-se a apresentar alguns conceitos por trás da abordagem utilizada pela biblioteca.

O spaCy, biblioteca com amplas aplicações em processamento de linguagem natural no geral, tem sua própria biblioteca de *deep learning* intitulada Thinc AI, utilizada ”por baixo dos panos“ para diferentes modelos de PLN. Para a maioria das tarefas, o spaCy usa uma rede neural profunda baseada em CNN (do inglês *Convolutional Neural Network*

[3] https://spacy.io/

). Especificamente para o Reconhecimento de Entidade Nomeada (do inglês *Named Entity Recognition*), o spacy usa:

1. Uma abordagem baseada em transições, descrita no artigo "Arquiteturas neurais para reconhecimento de entidades nomeadas"por (LAMPLE, 2016).
2. Uma estrutura chamada "Incorporar. Codificar. Participar. Prever":
    - Incorporar: as palavras são incorporadas usando um filtro *bloom*, o que significa que as *hashes* de palavras são mantidas como chaves no dicionário incorporado, em vez da palavra em si. Isso mantém um dicionário de incorporação mais compacto, com palavras potencialmente colidindo e terminando com as mesmas representações vetoriais.
    - Codificar: A lista de palavras é codificada em uma matriz de frases, para levar em consideração o contexto. O spaCy usa uma CNN (rede convolucional) para codificação.
    - Atender: Decida quais partes são mais informativas em uma consulta e obtenha representações específicas do problema.
    - Prever: spaCy usa um *perceptron* (uma unidade em redes neurais, um neurônio artificial) multicamada para inferência.

Além disso, spaCy fornece uma funcionalidade de *pipeline* que abstrai o treinamento e uso do classificador. O pipeline padrão é definido em um arquivo JSON associado ao modelo instanciado e que está sendo utilizado. Um dos componentes de *pipeline* que o spaCy fornece é chamado *textcat* (abreviação de TextCategorizer), que permite atribuir categorias (ou rótulos) aos seus dados de texto e usá-los como dados de treinamento para uma rede neural. Esse processo gerará um modelo treinado que poderá ser utilizado para predição de novos textos.

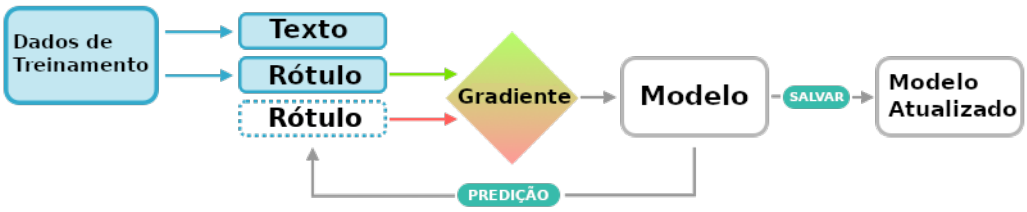


Figura 7 – Fluxo de Treinamento na *pipeline* do spaCy

A figura 7 ilustra o processo iterativo de treinamento e atualização do modelo. Os valores de peso utilizados são estimados com base em exemplos que o modelo observou durante o treinamento. O treinamento é um processo iterativo no qual as previsões do modelo são comparadas a fim de determinar o gradiente (losango ao centro da imagem).

Os gradientes indicam como os valores de peso devem ser alterados para que as previsões do modelo se tornem mais semelhantes aos rótulos de referência ao longo do tempo. O modelo (atualizável) obtido pode então ser utilizado para classificar novas instâncias com base nos pesos obtidos em seu treinamento.

3 Trabalhos Relacionados

Devido ao crescente interesse na descoberta de conhecimento em redes sociais, muitas linhas de pesquisa têm sido desenvolvidas nesse campo. Ainda que não tenha sido identificado na literatura uma proposta de ferramenta visual que auxiliasse a automação de todas as etapas do processo (desde o agendamento da coleta até a análise), outros trabalhos lidaram com desafios parecidos, seja na análise do discurso ou da rede de usuários e, portanto, oferecem boas referências para análise e comparação.

O trabalho desenvolvido por (BENEVENUTO, 2011), por exemplo, destrincha de forma semelhante (embora apenas no âmbito teórico) todo o processo desde a coleta em redes sociais até a análise dos dados obtidos. Através da aplicação de métodos baseados na teoria de redes complexas, o trabalho demonstra como obter métricas que auxiliam no entendimento da formação, estrutura e comportamento da rede. Probabilidade de agrupamento entre dois vértices, densidade, usuários(nós) mais centrais de acordo com diferentes métricas são algumas dessas possíveis análises. Entendendo como esse processo de análise é realizado, o trabalho aqui desenvolvido pode apoiar as etapas que podem ser automatizadas.

A fim de sumarizar a polaridade da opinião dos usuários, uma das funcionalidades de maior interesse nesse contexto, se trata da análise de sentimento na base coletada. O trabalho desenvolvido por (ARAÚJO, 2013) se propôs a comparar a eficácia de algoritmos disponíveis na literatura com relação a determinação de polaridade (sentimento positivo ou negativo) para uma base pré-rotulada. Para os métodos investigados, o que obteve melhor acurácia foi o de *emoticons* (que busca identificar conjuntos de caracteres representando emoções que tornem evidente o sentimento do texto). Um problema, identificado na própria análise do trabalho, é a abrangência do método, visto que uma amostra muito pequenas dos conteúdos publicados possuem esses caracteres que expressam emoção.

Outro trabalho aplicado à análise de sentimento em contexto semelhante foi desenvolvido por (PESSANHA, 2020), realizando um estudo dos impactos da pandemia do COVID-19, bem como a repercussão do isolamento social efetuado nesse período. Nesse estudo adotou-se a abordagem léxica (técnica não-supervisionada) para a verificação da polaridade (positivo e negativo) das opiniões. Assim como outros trabalhos de análise em língua portuguesa, essa pesquisa utilizou-se do dicionário léxico SentiLex-PT02. Esse método, devido à facilidade de implementação, chegou a ter sua viabilidade testada neste trabalho, mas foi descartado devido à baixa acurácia na classificação.

A abordagem empregada no trabalho aqui desenvolvido, por fim, de forma semelhante a (LIMA; CASTRO, 2012), fez uso da vantagem da presença de *emojis* (caracteres

que indicam emoção) nas publicações para inferir previamente o sentimento. Ainda que haja uma baixa cobertura de *tweets* com *emojis* presentes para inferir diretamente o sentimento do conteúdo, aqueles que possuem podem ser utilizados para treinamento de um classificador. A base de dados aqui utilizada[1] foi previamente rotulada seguindo esse critério, e sua escolha se deu principalmente por estar no mesmo contexto (política) e idioma (português brasileiro) da proposta da aplicação.

Na tentativa de identificar usuários com comportamento não-autêntico (implicando em potenciais contas falsas - ou robôs), um dos maiores empecilhos é fazê-lo utilizando os poucos atributos disponíveis, tanto em referência ao usuário quanto às publicações do mesmo. Entretanto, para alguns perfis onde não há um cuidado na replicação de um comportamento mais usual de um autor autêntico, esses atributos presentes na base podem indicar o comportamento suspeito. Assim como o trabalho aqui desenvolvido, (LEITE, 2019) utilizaram a estratégia de computar uma pontuação de autenticidade através de regras baseadas nos metadados coletados. Todavia, ao utilizarem diretrizes distintas, dentre elas algumas fazem uso de atributos extras que devem ser minerados (inviabilizando a utilização em larga escala, devido ao limite de requisições da API do Twitter).

Na tratativa de redes complexas compostas pelos usuários autores das postagens, o volume de nós e arestas pode (a depender do volume coletado) eventualmente inviabilizar o cálculo de métricas e indicativos da topologia da rede. O algoritmo de *backbone* proposto por (SERRANO M. BOGUÑÁ, 2009) foi utilizado com o intuito de obter e trabalhar uma subamostra significativa da rede. Através do filtro de disparidade calculado para as arestas do grafo, é possível preterir o subconjunto de vértices pertencentes à mesma como uma rede mais viável (que representa a "espinha dorsal", que nomeio o algoritmo) para detecção de comunidades e análise da topologia.

Ainda na análise de redes complexas que representam interações entre usuários, outros autores se propuseram a estudar o comportamento desses atores em cenários semelhantes. Os autores (BARBOSA, 2022) utilizaram as fontes de conteúdo do usuário (baseado nos domínios dos *links* disseminados) para caracterizar as comunidades de acordo com as referências utilizadas, também no contexto de COVID-19 (entretanto com foco no tópico de vacinação). (BARBOSA, 2019) desenvolveu também na área de redes sociais um *framework* para entender a propagação de conteúdos baseados em eventos reais.

[1] https://www.kaggle.com/datasets/augustop/portuguese-tweets-for-sentiment-analysis

4 Materiais e métodos

Durante a implementação dos algoritmos e das funcionalidades presentes na plataforma, algumas decisões de projeto, arquitetura e implementação foram tomados levando em conta o contexto e finalidade da ferramenta. Este capítulo destina-se a elaborar e detalhar esses métodos e justificar suas escolhas.

4.1 Arquitetura da Aplicação

O primeiro passo, fundamental para as escolhas subsequentes, foi definir o uso das tecnologias, *frameworks*, bibliotecas e demais recursos que permitiriam a implementação da plataforma. Esta seção é destinada a detalhá-los dentro do arcabouço da aplicação e como eles relacionam entre si. A Figura 8 representa visualmente uma abstração de como essas interações podem ser modeladas no fluxo da aplicação.

É possível verificar, a alto nível, um conjunto de ferramentas no grupo da direita voltados para execução e apoio diretamente à aplicação. O grupo à esquerda, composto por bibliotecas e scripts em Python, voltados majoritariamente para cálculos e computações de métricas e *scores* a serem utilizados pela aplicação. Ambos os grupos se comunicam com o banco de dados compartilhado, que executa sobre o SGDB MongoDB. Nas subseções que seguem, são detalhadas a proposta geral de cada tecnologia e sua finalidade particular no contexto em que foi aplicada.

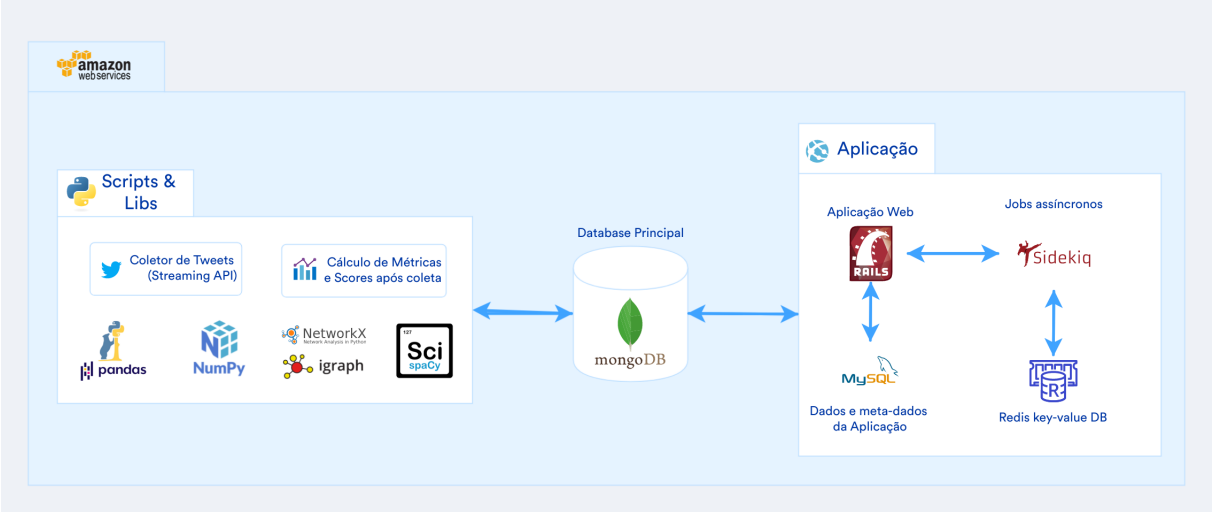


Figura 8 – Arquitetura da Aplicação e Processos Relacionados

4.1.1 MongoDB - Base de Dados Principal

O MongoDB é um conhecido Sistema Gerenciador de Banco de Dados (SGBD) não-relacional, também referenciado como um SGDB NoSQL. A implicação conceitual desse modelo, em contra-ponto aos SGDB's relacionais tradicionais, é não-existência de uma estrutura fixa (geralmente abstraída como uma tabela), mas ser orientado a documentos no formato chave-valor (de forma muito similar à conhecida estrutura de um JSON (*JavaScript* Object Notation). Essa flexibilidade permite lidar melhor com objetos complexos que precisem lidar com a possível presença/ausência de muitos atributos (evitando uma tabela esparsa e que precise de constantes atualizações).

No contexto da aplicação, uma vez que os objetos principais a serem trabalhados são Tweets que possuem essas mesmas características, o MongoDB se torna uma solução trivial para lidar com essa estrutura flexível dos objetos. No objeto retornado pela API do Twitter, constam não somente informações do tweet coletado em si, mas também alguns outros meta-dados referentes ao autor, geolocalização e outras informações relevantes (dentre as quais, alguns atributos podem ou não estar presentes). O grande volume de dados também é um fator relevante na opção por um banco de dados não-relacional: no caso de uma modelagem relacional, essas informações precisariam ser coletadas em diferentes tabelas e mescladas via uso de *joins* no SQL (o que pode ser oneroso à medida que o volume de dados cresce). Na abordagem não-relacional, "*o que é utilizado em conjunto, é armazenado em conjunto*"(ponto importante de atenção na modelagem que utiliza este paradigma).

Como todos os atributos presentes no objeto poderiam ser relevantes e utilizados em algum momento por alguma funcionalidade ou método da aplicação, optou-se por armazenar integralmente o objeto, sem alteração/pré-processamento nos valores retornados. Dessa forma, a manipulação e tratamento desses dados é uma responsabilidade transferida para a ponta (nas funcionalidades da aplicação).

4.2 Coleta de Tweets - *Streaming* API

A primeira etapa de todo processo a ser desenvolvido consiste na obtenção dos *tweets* (como são chamadas as publicações da rede) dentro do contexto a ser observado. A API do Twitter fornece duas abordagens que podem ser utilizadas: A *Search* API, que possibilita a obtenção de conteúdos retroativos segundo palavras-chaves de interesse (dentre outras opções da parametrização) através de uma REST API. A outra opção, aqui utilizada, é a *Streaming* API, que por sua vez permite a "escuta"em tempo de execução sobre um conjunto de termos de interesse.

A escolha é justificada dado o interesse em coleta de eventos em tempo real durante uma janela de tempo, abordagem mais próxima da *Streaming* API. Além disso, nesta, em

vez de entregar dados em lotes por meio de solicitações subsequentes do aplicativo cliente (como no caso de uma API REST), uma única conexão é aberta entre o aplicativo e a API, com novos resultados sendo enviados por meio dessa conexão sempre que ocorrerem novas correspondências. Isso resulta em um mecanismo de entrega de baixa latência que pode dar suporte a uma taxa de transferência mais alta.

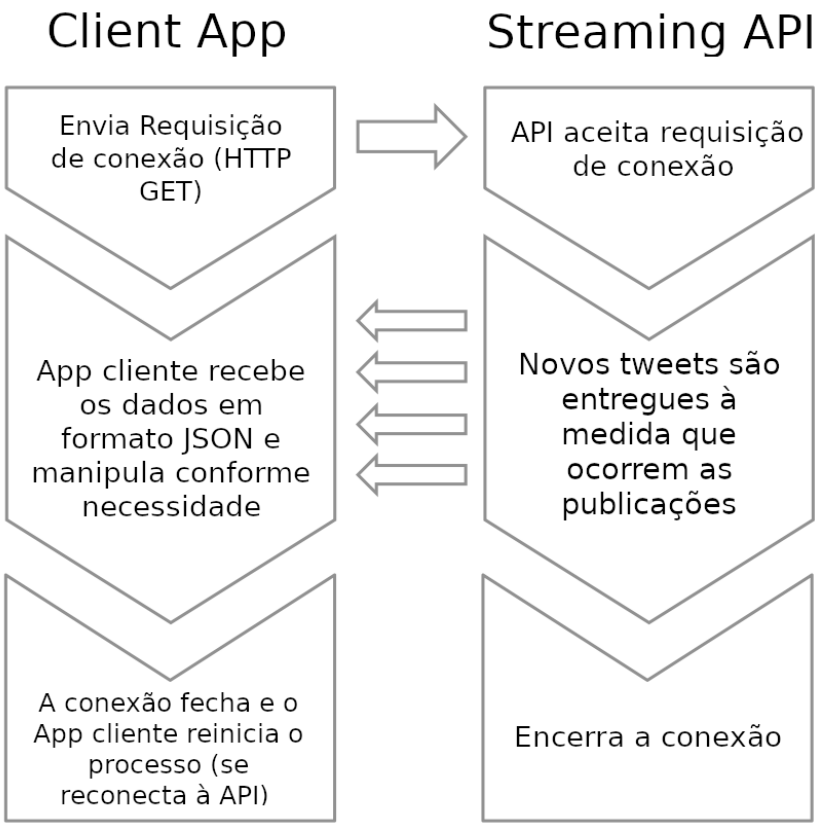


Figura 9 – Fluxo do consumo da API de *Streaming* <Disponível em https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data>

A Figura 9 ilustra como é realizado o fluxo de consumo de conteúdos através dessa API. Essa implementação foi auxiliada pela utilização da biblioteca Tweepy (descrita na subseção a seguir), que simplifica a utilização, tornando necessário apenas a definição dos *tokens* de acesso, parâmetros de busca e a manipulação dos objetos retornados na escuta. O código 2 demonstra a implementação dos método que recebe os objetos e os insere no banco (*on_status(self,tweet)*). É possível verificar que os *tweets* com o atributo "*possibly_sensitive*"são rejeitados, uma vez que podem conter conteúdo impróprio.

No caso de erro, cuja responsabilidade de tratamento é da função *on_error(self, status_code)*, a descrição do erro é exibida e o coletor entra em *sleep* por 15 minutos (tempo de janela considerado pela API). Como cada coletor agendado possui um ID referente ao *job* responsável por sua execução, a plataforma tem agendado (de acordo com o período de coleta) o comando de finalização do processo assim que atingido o período limite. Em seguida, o *job* responsável pelo cálculo de métricas e *scores* (centralidades,

```
class CustomStreamListener(tweepy.StreamListener):

  def on_status(self, tweet):

    try:
      if not hasattr(tweet, 'possibly_sensitive'):
        db.tweets.insert_one(tweet._json)

    except tweepy.TweepError:
      time.sleep(60 * 15)

    return True

  def on_error(self, status_code):
    print "Erro com o código:", status_code
    return True # Não mata o coletor
```

Listing 2: Classe customizada com implementação dos métodos do Tweepy que lidam com os objetos retornados na escuta

análise de sentimento, autenticidade) é executado.

4.2.1 Scripts e Bibliotecas em Python

A linguagem Python é amplamente utilizada (mantendo uma crescente nos últimos anos) ao se trabalhar com dados em aplicações de mineração, tratamento, manipulação e demais aplicações em geral na área de ciência de dados (COURSERA, 2022). Grande parte dessa notoriedade é justificada por suas bibliotecas de boa performance e fácil utilização (mesmo para usuários que não possuam um conhecimento avançado em programação). No contexto da aplicação e pesquisa aqui desenvolvidos, foram implementadas algumas rotinas necessárias para a execução processos relacionados à obtenção de algumas informações. Abaixo detalha-se como essas bibliotecas apoiaram essas implementações:

- **Tweepy**[1]: A biblioteca Tweepy visa facilitar e automatizar as interações com as API's do Twitter. Foi utilizada no processo de escuta, coleta e armazenamento dos Tweets relativos aos termos-chave de interesse.
- **Pymongo**[2]: A biblioteca Pymongo é um *driver* em Python que permite interagir diretamente com o SGDB não-relacional MongoDB. Portanto, é utilizado na grande maioria dos scripts, seja para inserção dos dados na coleta, ou na recuperação dos mesmos em computações subsequentes.

[1] https://www.tweepy.org/
[2] https://pymongo.readthedocs.io/en/stable/

- **Pandas**[3] **& Numpy**[4]: Ambas surgiram como bibliotecas essenciais para qualquer computação científica em aplicações que incluem manipulação de vetores e matrizes, métodos estatísticos e aprendizado de máquina. Na aplicação, foram utilizadas principalmente para análise de sentimento e modelagem de tópicos.

- **Igraph**[5] **& NetworkX**[6]: Bibliotecas utilizadas para aplicações em grafos e redes complexas em geral. Possuem diversos algoritmos para cálculos de métricas e indicadores de uma rede, e foram utilizadas principalmente para detecção de comunidades (NetworkX) e medidas de centralidade (Igraph).

- **SpaCy**[7]: Utilizada para aplicações em Processamento de Linguagem Natural (PLN) no geral, o spaCy foi utilizado em tarefas preliminares como comuns de pré-processamento de texto, como *tokenização*, *lemmatização*, *stemmatização* e remoção de *stop words*. Obtidos os conjuntos de *tokens* pré-processados, apoiou também na análise de sentimento cuja abordagem consiste no algoritmo de redes neurais para treinamento e classificação de sentimento dos tweets.

- **Scikit Learn**[8]: O Scikit-learn (anteriormente scikits.learn e também conhecido como sklearn) é uma biblioteca de aprendizado de máquina *open-source*. Ela disponibiliza vários algoritmos de classificação, regressão e agrupamento. Foi projetada para operar em integração com as bibliotecas numéricas e científicas NumPy[9] e SciPy[10]. Foi utilizada para implementar a modelagem de tópicos (tanto no método NMF quanto no LDA).

4.2.2 Aplicação Principal

A aplicação principal, a qual o usuário efetivamente tem acesso e interage para configuração, execução e visualização dos resultados, é executada como uma plataforma web (dispensando instalação/atualização em máquina cliente e sendo compatível com qualquer dispositivo que execute um navegador web). Para desenvolvimento dessa aplicação, foi utilizado o *framework web* de desenvolvimento ágil Ruby On Rails[11], que possui fácil integração com soluções de autenticação de usuário, consultas ao MongoDB, dentre outras funcionalidades presentes na aplicação. Para hospedagem da aplicação e serviços relacionadas à mesma, foram utilizados servidores em nuvem da Amazon AWS.

[3] https://pandas.pydata.org/
[4] https://numpy.org/
[5] https://igraph.org/
[6] https://networkx.org/
[7] https://spacy.io/
[8] https://scikit-learn.org/
[9] https://numpy.org/
[10] https://scipy.org/
[11] https://rubyonrails.org/

Para armazenar informações relativas à aplicação (como dados de usuário, informações de coletas agendadas, dentre outros-metadados), foi utilizado o SGDB relacional MySQL[12]. Diferentemente da base principal da aplicação, responsável por armazenar os tweets, para a base da aplicação, os dados são estruturados, de baixo volume, e integram melhor com um SGDB relacional. Dessa forma, todos os dados que são referentes à aplicação em si, são persistidos nesse banco de dados, que executa no mesmo servidor da aplicação.

Outra tarefa importante a ser executada pela aplicação, são as coletas e outras rotinas que devem ser realizadas de forma assíncrona, uma vez que o usuário vai fazer o agendamento para determinado horário e encerrar sua interação com o sistema. Para essa execução em *background*, foi utilizado o Sidekiq[13], responsável por agendar, gerenciar e executar esses *jobs* em Ruby. O Sidekiq, por sua vez, utiliza o Redis[14], um banco de dados do tipo chave-valor, para gerenciar a sua fila de tarefas, bem como os seus respectivos parâmetros.

A aplicação faz ainda, direta ou indiretamente, a utilização dos *scripts* e bibliotecas em Python mencionadas na subseção anterior. Para diversas funcionalidades, o uso direto se dá por meio da utilização do retorno desses *scripts* para verificar, por exemplo, quais usuários pertencem a uma determinada comunidade. Para outros casos, como na consulta e comparação de métricas e *scores*, esse valor é lido diretamente do MongoDB, após processamento e armazenamento para consulta direta. Esta prática evita retrabalho em termos de processamento uma vez que, no contexto da aplicação, é trabalhada uma base de dados de fotografia fechada, sem atualização contínua e, portanto, sem alteração nesses valores.

4.3 Funcionalidades

4.3.1 Agendamento de Bots (Robôs de Coleta)

Funcionalidade para cadastro/programação de coleta de *tweets* informando a hora de início, hora final, termos a serem coletados e base a ser armazenada. O processo é executado em *background* pelo Sidekiq e as bases/*collections* têm escopo por usuário. Na etapa inicial de todo o fluxo, a tarefa primordial da coleta de Tweets foi apoiada pela biblioteca Tweepy[15], que automatizou o processo de estuca da Streaming API[16] do Twitter. Essa API, diferente da API de busca (que retorna resultados retroativos em lotes), habilita um modo de "escuta"baseado em palavras-chaves de interesse. Durante o

[12] https://www.mysql.com/
[13] https://github.com/mperham/sidekiq
[14] https://redis.io/
[15] https://www.tweepy.org/
[16] https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data

período de escuta, uma fração dos Tweets relacionados são obtidos e podem ser utilizados de acordo com a necessidade da aplicação.

4.3.2 Dashboard principal

A tela inicial da aplicação, após efetuado o login, é um painel com as principais informações consolidadas da coleta efetuada. Essa *dashboard* principal contém informações a respeito do período de coleta, número de *tweets*, número de autores únicos e as funcionalidades implementadas: consolidação do sentimetno positivo/negativo ao longo da coleta, mapa de calor da localização e sistemas operacionais dos usuários, série temporal da análise de sentimento e uma nuvem de palavras com os termos mais utilizados. Essas visualizações foram apoiadas pelo uso de bibliotecas externas, como o ChartJS[17], o *Magic Cloud*[18] e o Google Maps[19]. Através da incorporação em colunas responsivas do Bootstrap[20], foi possível incoroporar esses elementos de forma responsiva à aplicação.

Para recuperação dos dados que alimentam a tela inicial, bem como outros métodos relacionados à consultas no banco de dados MongoDB, foi implementada a classe MongoHelper, cujo esqueleto e assinatura dos métodos estão descritos no código 3. É possível observar que boa parte dos métodos recebem como parâmetro opcional de lista de usuários. Caso esse parâmetro esteja presente, a busca e o retorno são delimitados ao escopo daquela lista de usuários. Essa decisão de implementação foi tomada pela necessidade de reutilização dos métodos para escopos de comunidades de usuários, nas quais as mesmas visualizações estão presentes, com domínio limitado ao grupo exibido.

4.3.3 Dashboard de tendências - *What's Happening*

Subjacente à *Dashboard principal*, com o intuito de analisar eventos relacionados e tendências ocorridas no mesmo período da coleta, foi implementado o painel "*What's Happening*- "o que está acontecendo", que integra algumas funcionalidades como o *google trends*, mapa de interesse, *twitter trends* e busca utilizando a *Search API* do Twitter. Para todas elas, é segmentado a janela de tempo de acordo com a coleta, e os termos a serem filtrados de acordo com os termos da busca também utilizados para coleta. A incorporação dessas funcionalidades em uma aplicação web se dá de forma trivial através de objetos embutidos em *iframes*, nos quais as URL's contém os parâmetros para a segmentação descrita.

Além desses, foi implementada também uma listagem de notícias utilizando o portal de notícias G1[21]. Neste caso, na ausência (pelo menos até o momento da imple-

[17] https://www.chartjs.org/
[18] https://github.com/zverok/magic_cloud
[19] https://cloud.google.com/apis
[20] https://getbootstrap.com/
[21] https://www.g1.globo.com

```ruby
class MongoHelper

    def get_non_authentic_users
           offset, limit, sort_column, sort_direction, search
        )
    end

    def load_map_points(users=nil)

    end

    def operational_systems_data(users=nil)

    end

    def overall_sentiment(users=nil)

    end

    def sentiment_time_series(users=nil)

    end

    def domain_links(users=nil)

    end

    def word_cloud_data(users=nil)

    end

    def influent_users(criteria, limit, normalize, users=nil)

    end

    def time_series_collection

    end


end
```

Listing 3: *Helper* em Ruby para recuperação de informações no MongoDB

```ruby
require 'open-uri'

class WebScrapper

    def self.g1_noticias(search)

        url = "https://g1.globo.com/busca/?q=#{CGI.escape(search)}"

        page = open(url) { |f| Nokogiri::HTML(f) }
        page.css(".results__content .widget")[1..-2].to_a.map{|n|
            {
                img: n.css("img").first.to_s,
                title: n.css(".widget--info__title").first.
                       try(:text).to_s.gsub("\n","").squeeze.strip,
                date: n.css(".widget--info__meta").try(:text),
                link: (n.css(".widget--info__text-container").
                       css("a").last['href'] rescue "")
            }
        }.select{|e| e[:title].present?}

    end

end
```

Listing 4: Código em Ruby da implementação do WebScrapper que obtém as notícias do G1

mentação) de um *iframe* ou API para obtenção dos dados, foi utilizado um *web scrapper*, algoritmo responsável por fazer uma chamada à página web de busca e tratar os dados em formato não estruturado (HTML) para exibição junto às demais funcionalidades. A implementação foi auxiliada pela biblioteca Nokogiri[22] (em Ruby), que auxilia na navegação e recuperação de informação em páginas web. A classe implementada é descrita no código 4 e seu método principal recebe como parâmetro os termos a serem buscados.

4.3.4 Modelagem de Tópicos (NMF)

Como a modelagem de tópicos não requer treinamento, é um bom ponto de partida para começar a analisar dados textuais, como os *tweets* coletados. O método descrito no código 5 descreve as principais etapas necessárias desde a obtenção dos dados até a sua visualização, cuja saída é uma lista com os 10 principais tópicos. Assim como as demais funcionalidades que tratam de linguagem natural, o pré-processamento realizado na base visa normalizar os termos e possibilitar uma análise mais assertiva. A implementação do

[22] https://nokogiri.org/

NMF (assim como o LDA) foi realizada com o auxílio da biblioteca *Scikit Learn*[23]. A implementação do método principal descrita no código tem seus passos melhor descritos nos parágrafos a seguir.

Antes de tudo, o *dataframe* passado por parâmetro (df) já contém uma coluna *text_preprocessed*, a qual já contem a saída do pré-processamento utilizado nas aplicações de PLN. A variável *tfidf_vectorizer*, instanciada utilizando a classe do *Scikit Learn*, recebe os seguintes parâmetros:

- **stop_words**: lista de termos identificados a priori como *stopwords* do idioma e que devem portanto, ser removidas. Na aplicação, foi utilizada a lista padrão de *stopwords* provida pela biblioteca spaCy.
- **max_df**: ignora palavras que são comuns entre os documentos e não agregam um grande significado. Esse valor pode ser absoluto ou relativo (razão). Com o intuito de remover termos comuns aos documentos (potenciais *stopwords* não identificadas na lista anterior) foi utilizado um limiar de 0.95, ou seja, palavras que aparecem em mais de 95% dos documentos.
- **min_df**: ignora os documentos que aparecem menos que o valor do parâmetro. Neste caso, dado o volume de *tweets*, espera-se que um termo importante ocorra em mais de 1 documento, pelo menos. Portanto, optou-se pelo número absoluto 2, ou seja: termos que aparecem em pelo menos 2 *tweets*.
- **ngram_range**: é o limite inferior e superior de palavras a serem extraídas. No contexto da aplicação, os tópicos são extraídos em pares (ou bigramas), ou seja, um limite inferior/superior de (2,2).

Obtido o resultado, os termos no vetor são classificados em ordem decrescente de acordo com a pontuação obtida. No código exemplificado, os 10 tópicos mais relevantes são exibidos para inspeção. Já na aplicação, a saída completa da lista de termos por NMF foi utilizada na criação de uma nuvem de palavras sobre a bandeira do país, demonstrada nos resultados na figura 12. Além desta, uma outra visualização presente na aplicação foi desenvolvida com o intuito de exibir a lista de termos referentes a cada tópico individualmente, bem como um indicativo da probabilidade visualmente indicada do termo pertencer ao tópico. Essa visualização, por sua vez, foi implementada utilizando a modelagem de tópicos por LDA, e é descrita na subseção a seguir.

[23] https://scikit-learn.org/stable/index.html

```
def topic_model(df):
  # executando o tf-idf
  tfidf_vectorizer = TfidfVectorizer(
     stop_words=stop_words, max_df=0.95,
     min_df=2, ngram_range=(2,2)
   )
  tfidf = tfidf_vectorizer.fit_transform(
    df.loc[:, 'text_preprocessed']
  )
  # classicando a pontuação
  sorted_items=sort_coo(tfidf[0].tocoo())
  # relacionando as palavras, classificado pela posição na matriz
  feature_names_tfidf=tfidf_vectorizer.get_feature_names()
  n_grams_tfidf=extract_topn_from_vector(
    feature_names,sorted_items,10
  )
  # instanciando o NMF
  nmf_tf = NMF(10)
  # treinando o tfidf para NMF
  nmf_topics1 = nmf_tf.fit_transform(tfidf)
  top_words_per_topic(nmf_tf, tfidf)
```

Listing 5: Método principal da modelagem de tópicos utilizando o NMF com TF-IDF

4.3.5 Modelagem de Tópicos (LDA)

Assim como na implementação do NMF, a modelagem de tópicos por LDA foi apoiada pelo uso da biblioteca *Scikit Learn*. Os parâmetros *database* e *collection* no método determinam, respectivamente, o banco de dados e coleção de *tweets* no MongoDB a serem considerados na execução. Assim como no método anterior, a etapa de pré-processamento dos textos é realizada antes de fornecer a lista de documentos ao método *lda_scikit*. Este, além da lista, recebe como parâmetro também o número de tópicos a serem obtidos. Seguindo o valor padrão do método, e também visando uma melhor diagramação da visualização do resultado, 10 tópicos foram utilizados na execução. A visualização do resultado, por sua vez, é auxiliada pela plotagem da biblioteca Matploblib [24]. Nessa plotagem, o tamanho da barra referente a cada termo é ponderada pelo seu peso dentro do tópico referente.

4.3.6 Análise de Sentimentos

A análise de sentimentos foi implementada com auxílio da biblioteca de PLN (processamento de linguagem natural) spaCy[25], cujo modelo de predição foi alimentado por

[24] https://matplotlib.org/
[25] https://spacy.io/

```
def main(collection, database):
    """Run topic model."""

    processing = PreProcessing()
    tweet_filter = {"lang": "pt"}
    projection = {'text': 1,
                  'extended_tweet.full_text': 1,
                  'retweeted_status.text': 1,
                  'retweeted_status.extended_tweet.full_text': 1}

    # Recupera tweets do banco no MongoDB
    tweets = storage.get_filtered_tweet(tweet_filter, projection, 0)

    # Extrai o texto dos tweets
    raw_texts = list(map(get_full_text, tweets))

    # Limpa o texto
    cleaned_texts = list(map(processing.clean_text, raw_texts))

    # realiza a modelagem de tópicos
    model_lda, names = lda_scikit(cleaned_texts, 10)

    # Exibe os tópicos
    plot_top_words(model_lda, names, 10, "", "blue")
```

Listing 6: Método principal da modelagem de tópicos em LDA, responsável por recuperar no banco, fazer o processamento e gerar a visualização em imagem

uma base pré-classificada disponível no Kaggle[26], cujo contexto é o mesmo da aplicação: *tweets* em português para análise de sentimento. Com a utilização de uma base já rotulada para treinamento, o modelo obtido é utilizado para classificação dos conteúdos coletados pela aplicação.

Uma vez que o modelo, após treinado, pode ser salvo em disco para carregamento ágil em utilizações posteriores, esse treinamento foi realizado para toda a base, que dispunha de 50 mil *tweets*, divididos igualmente entre conteúdos de teor negativo e positivo. Após o fim da coleta, a aplicação realiza a computação do *score* de sentimento de cada *texto* e o armazena no banco, uma vez que esse valor não se altera e armazená-lo permite a sua recuperação sem a necessidade de recálculo para uso na funcionalidade da plataforma.

[26] https://www.kaggle.com/datasets/augustop/portuguese-tweets-for-sentiment-analysis

4.3.7 *DataTables* - Listagem de Tweets e Usuários não-autênticos

A fim de permitir que o usuário fizesse uma consulta em toda a base coletada, bem como ordernar, buscar e ver meta-dados dessa base (como o número total de registros), a biblioteca DataTable[27] (implementada na linguagem JavaScript sobre o *framework* JQuery[28]) foi utilizada, uma vez que já possui todas essas funcionalidades implementadas. O necessário para sua utilização consiste em prover os dados para o *front end*, onde a *Data Table* é renderizada. Em aplicações usuais, a tabela poderia ser carregada integralmente, de forma síncrona. Mas dado o volume esperado de dados, o ideal é utilizar uma chamada assíncrona ao servidor, a qual contém nos seus parâmetros a página atual, número de registros por página, coluna de ordenação, dentre outras informações, de forma que o servidor deve receber esses parâmetros e fornecer a resposta de acordo com a requisição.

Como a resposta possui um formato pré-definido e a funcionalidade é implementada também na listagem de usuários não-autênticos seguindo o mesmo princípio, foram implementadas três classes responsáveis por essas tratativas. Uma superclasse (nomeada *TemplateDatatable*), descrita no O código 7, trata e parametriza a resposta de forma genérica o suficiente para serem utilizadas pelas subclasses. Estas, por sua vez, tem a responsabilidade de implementar os métodos que efetivamente buscam os valores (no caso da aplicação, no banco de dados do MongoDB) e os mapeiam nas colunas esperadas a serem exibidas. O código 8 exemplifica o corpo da classe responsável pelo retorno da listagem de Tweets. Com essa estrutura genérica, indeterminadas novas *data tables* podem ser implementadas pela reutilização de código coeso.

4.3.8 Potenciais perfis não-autênticos

Conforme mencionado no referencial teórico, das diretrizes propostas por Camizani-Calzolari (CRESCI, 2015), uma parcela não pode ser verificada no retorno padrão da *Streaming API* do Twitter, pois requer a mineração de dados adicionais (como coleta do *feed* do usuário, lista de seguidores, dentre outros). Uma vez que essa coleta se torna inviável para uma base maior (dado o limite de requisições da API), somente aquelas podem ser verificadas na proposta do autor foram consideradas. Além disso, foram utilizados pesos de acordo com a relevância da diretriz. O conjunto final das diretrizes observáveis que determina o score (inicialmente zerado) é computado conforme descreve a lista que segue:

- Nome presente: +1
- Imagem de perfil presente: +1
- Localidade presente: +1

[27] https://datatables.net/
[28] https://jquery.com/

```ruby
class TemplateDatatable

  #View_context para acessar dados da sessão
  def initialize(view, db=nil)
    @view = view
    @db = db
  end

  def as_json(options = {})
    {
      sEcho: params[:sEcho].to_i,
      iTotalRecords: objects.size,
      iTotalDisplayRecords: objects.total_entries,
      aaData: data
    }
  end

private

  def objects
    @objects ||= fetch_objects
  end

  def page
    params[:start].to_i/per_page + 1
  end

  def per_page
    params[:length].to_i
  end

  def sort_column
    order_column = params.to_unsafe_h["order"]["0"]["column"].to_i
    sort_columns_collection[order_column]
  end

  def sort_direction
    params.to_unsafe_h["order"]["0"]["dir"]
  end

end
```

Listing 7: Superclasse TemplateDatatable, com os principais métodos para renderização da tabela

```ruby
class TweetsDatatable < TemplateDatatable


private

  def data
    objects.map do |tweet|
      Tweet::VIEW_COLUMNS.keys.map{|c|
        #Mapeamento das colunas a serem obtidas nos objetos
        # retornados pelo método fetch_objects
      }
    end
  end

  def fetch_objects
    MongoHelper.new(@db).load_collection(
      "tweets", page.to_i-1, per_page, sort_column,
      sort_direction,params[:search][:value]
    )
  end

end
```

Listing 8: Subclasse TweetsDatatable, mapeando os valores a serem retornados

- Descrição (bio) presente: +1
- Número de seguidores > 100: +1
- Presente na lista de pelo menos algum outro usuário: +1
- Tem pelo menos 100 *tweets*: +1
- URL (website) presente: +1
- Utiliza Twitter Web, Android ou iOS: +1
- O número de seguidores é pelo menos o dobro de quem está seguindo: +1
- A conta existe há pelo menos 5 anos: +1
- A descrição (bio) contém o termo "bot": -1
- A quantidade de números no screen_name é maior do que 5: -1
- O número de seguidores é pelo menos 50 vezes menor do que está seguindo: -3

Uma vez que, dada a natureza da aplicação que consiste de uma análise em um intervalo de tempo, é possível obter uma fotografia ao final da coleta que reflita esses valores/*scores* computados na base. Ou seja, como os atributos e valores coletados não irão ser alterados, isso implica na possibilidade de calcular o índice de autenticidade de cada usuário que gerou conteúdo na base e armazená-lo como um atributo adicional, de forma a possibilitar sua recuperação imediata e facilitar a ordenação ou outras operações.

Baseado nesse *score* obtido, a lista de potenciais perfis não-autênticos consiste basicamente em definir um limiar de corte a ser aplicado sobre a pontuação final computada. Nos resultados obtidos, foi verificado (através de uma inspeção qualitativa) que a utilização de um *score* mais conservador (de pontuação final menor ou igual a 1) filtrou com boa segurança e assertividade os potenciais perfis duvidosos. Na plataforma, essa visualização está disponível através de uma *DataTable* e, assim como na listagem de *tweets*, os resultados podem ser paginados, ordenados e pesquisados de acordo com os atributos da tabela.

4.3.9 Usuários mais centrais/influentes

Conforme descrito na seção de arquitetura da aplicação, dentre os *scripts* e bibliotecas em Python, o Igraph foi responsável por auxiliar no cálculo das métricas de centralidade dos nós (usuários) na rede. Para tal, o primeiro passo a ser considerado é a modelagem da rede. A abordagem mais trivial seria considerar uma relação de um usuário u_1 para um usuário u_2 caso o primeiro siga o segundo. Entretanto, para tal, seria necessário minerar toda a lista de amigos e seguidores de cada usuário da rede, o que é inviável pelo limite de requisições da API. Portanto, outra abordagem foi utilizada: a relação (direcionada) existe de um primeiro para um segundo usuário caso o primeiro tenha mencionado ou compartilhado algum conteúdo do segundo (o que é verificável na base de textos).

Definida a modelagem, ao fim do período de coleta, o script em Python responsável por varrer a base mapeando as relações no grafo é disparado. Após a varredura, as métricas de centralidade definidas (Grau, *Page Rank* e *Betweenness*) são calculadas e, dado que a coleta foi encerrada e portanto a rede não se modificará, esses valores podem ser armazenados no banco de dados do Mongo DB, referenciando para cada usuário, o seu *score* em cada métrica. Por fim, a funcionalidade de exibição desses usuários consiste apenas em fazer a consulta ordenada desses valores e exibí-los na página web.

4.3.10 Detecção e Análise de Padrões em Comunidades no *Backbone*

A funcionalidade que permite analisar os padrões e tendências em *clusters* (ou grupos) de usuários na base coletada decorre de dois processos na rede de usuários: a

extração do *backbone*, cuja saída é entrada para a detecção de comunidades de Louvain (ou seja, os grupos são considerados apenas na sub-rede de maior significância estatística da rede completa). Para tal, entretanto, é necessário definir a modelagem que irá gerar a rede complexa que reflete as interações entre os usuários da base. A abordagem mais trivial seria considerar um grafo direcionado no qual a existência de uma aresta entre um usuário u_1 para um usuário u_2 indica que o primeiro segue o segundo (já que no Twitter a relação não é necessariamente recíproca). Entretanto, tal abordagem faria necessário minerar toda a lista de seguidores e seguidos para cada usuário, encontrando também a inviabilidade do limite de requisições da API.

Na abordagem utilizada na aplicação optou-se por considerar uma condição verificável nos dados coletados. Essa condição define uma interação direcionada entre dois usuários no caso de menção em algum *tweet* (seja via re compartilhamento ou menção direta). Na definição formal o grafo G(V,E) que representa a rede de usuários é dada por:

- V é o conjunto de todos os usuários u que estão na base coletada;
- $E = \{\{u_1, u_2\} \mid$ u_1 menciona/compartilha o u_2 em pelo menos um *tweet*$\}$

Na execução dos métodos de *disparity filter* para obtenção do *backbone* e de Louvain para obtenção de comunidades, alguns parâmetros devem ser considerados segundo a própria definição dos algoritmos. Esses parâmetros podem ser alterados pelo usuário nos *inputs* da plataforma, mas tem o valor padrão sugeridos de acordo com a literatura. Também foram esses valores que foram consideradas na análise de resultados. Esses parâmetros são listados e descritos como:

- ***Resolution (Louvain)***: determina o favorecimento da obtenção de comunidades maiores ou menores, de acordo com o valor do parâmetro. Pare resoluções menores que 1, o algoritmo favorece a obtenção de comunidades maiores (com mais nós). Valores maiores que 1 favorecem a obtenção de comunidades menores. Valor padrão: 1.
- ***Randomize (Louvain)***: torna aleatória a ordem de avaliação dos vértices, podendo obter resultados diferentes a depender da ordem avaliada. Padrão: *False*.
- ***Alpha (Serrano)***: valor de corte para o qual a hipótese nula deve ser rejeitada no *disparity filter* (ou seja, arestas com valores menores ao limiar são retidas no *backbone*). Quanto maior seu valor, mais arestas são consideradas e portanto maior a rede obtida. Padrão: 0.1.
- **Min e Max de usuários na rede**: esse parâmetro não influencia no processo de obtenção do *backbone* ou das comunidades, mas é utilizado para filtro de forma que,

na visualização da plataforma, somente comunidades com o número de usuários no intervalo definido são exibidas. Padrão: 10 (min) e 99999 (max).

Na visualização da plataforma dos dados obtidos desse processo, a modularidade (como medida de qualidade a ser maximizada), bem como outros meta-dados (número de comunidades, média de usuários, dentre outros) são exibidos no cabeçalho da página. Além disso, a fim de viabilizar a interpretação e caracterização de cada comunidade, uma *dashboard* é exibida logo abaixo com as mesmas funcionalidades da *dashboard* principal, com a particularidade de que essas visualizações são limitadas ao escopo de cada comunidade. Um *select box* na página lista as comunidades obtidas (ordenadas decrescentemente pelo número de usuários) e permite a alteração da visualização de acordo com a opção selecionada.

4.3.11 Série temporal de Retweets populares

A série temporal de *retweets*, assim como os demais gráficos da aplicação, foi implementada com auxílio da biblioteca ChartJS, que possui uma funcionalidade específica para séries temporais (utilizada na análise de sentimento ao longo da coleta e na série de *tweets* mais populares). Para utilização da funcionalidade, foi necessário implementar na classe *helper* um método que filtrasse as informações utilizando o atributo *created_at* que indica a hora de postagem do conteúdo. Na visualização, os *tweets* mais populares (com maior número de compartilhamentos) são listados para o usuário em uma caixa de seleção, de forma que, ao selecionar determinado conteúdo, é refletido no gráfico a referente série temporal.

4.3.12 Exportação de Dados

A fim de permitir que o usuário exporte as informações coletadas para trabalhar em uma ferramenta externa, o sistema permite também a exportação dos objetos presentes na base coletada, no formato de planilha, auxiliado pela biblioteca em Ruby[29]. No ato de exportação, o usuário pode escolher os atributos que deseja exportar e obtém o arquivo no formato *xlsx*. Como todos os atributos são exportados, dados como do usuário (que contém mais atributos aninhados) possuem um formato não amigável e, portanto, como melhoria futura é desejável implementar a tratativa desses objetos.

[29] https://github.com/caxlsx/caxlsx_rails

5 Resultados

Este capítulo destina-se a apresentar visualmente, bem como também de forma analítica, os resultados obtidos pelas funcionalidades implementadas. O contexto da aplicação na coleta dos dados é de grande relevância na interpretação dos resultados apresentados e, consequentemente é requisito que o pesquisador (usuário da plataforma) tenha o *background* necessário para analisar corretamente as informações exibidas. Também a depender do cenário, os *insights* obtidos podem ser mais relevantes em alguma visualização específica. Dessa forma, as subseções seguintes exploram os principais padrões, tendências e características e conhecimentos que puderam ser observados com o auxílio da plataforma.

5.1 Dados coletados e meta-informações

Definido o evento de interesse (leitura do relatório da CPI do COVID) para estudo e validação da ferramenta e suas funcionalidades, a mesma foi configurada na plataforma para posterior análise. A coleta foi realizada apenas para *tweets* em português, utilizando os termos que tendiam a ser mais utilizados nas postagens (identificados através dos *trending topics*), bem como nomes de personagens relevantes no evento, tais como:

- #CPIdaCovid
- #CPIdaPandemia
- #CPIDaVergonha
- #CPIdoCirco
- #RelatorioDaCPI
- @MarcosRogerio
- @randolfeap
- @OmarAzizSenador
- @ContaratoSenado
- @SimoneTebetms

A coleta foi realizada entre os dias 20/10/2021 e 26/10/2021, período que contempla a leitura do relatório da CPI do COVID, bem como eventos e desdobramentos

```
{
"created_at": "Thu May 10 15:24:15 +0000 2021",
 "id_str": "850006245121695744",
 "text": "Here is the Tweet message.",
 "user": { ... },
 "place": { ... },
 "entities": { ... },
 "extended_entities": { ... }
}
```

Listing 9: Exemplo de Objeto JSON retornado com dados do *tweet*

relacionados à mesma. Na aplicação desenvolvida, o manuseio dos objetos retornados foram armazenados integralmente (ou seja, com todos os atributos retornados), a fim de se investigar e verificar aqueles que podem ser trabalhados e fornecer alguma informação relevante. Um exemplo de objeto retornado pela API pode ser verificado no trecho 9. A depender da limitação de compartilhamento de cada usuário, determinados atributos (como geolocalização, por exemplo) podem ou não estar presentes, se tratando de um formato não-estruturado (que é uma maleabilidade presente no MongoDB em possuir o *schema* flexível).

Durante o período de coleta, um volume total de 165.295 *tweets* foram armazenados, bem como seus meta-dados presentes no objeto. Dentre os autores, há um total de 60.267 usuários distintos (ou seja, em alguns casos um mesmo usuário publicou mais de uma postagem). Esses *tweets* coletados foram a fonte de dados para as funcionalidades implementadas na aplicação, cujos resultados são apresentados a seguir.

5.2 Funcionalidades e Visualizações Implementadas

5.2.1 Dashboard Principal e Informações Gerais

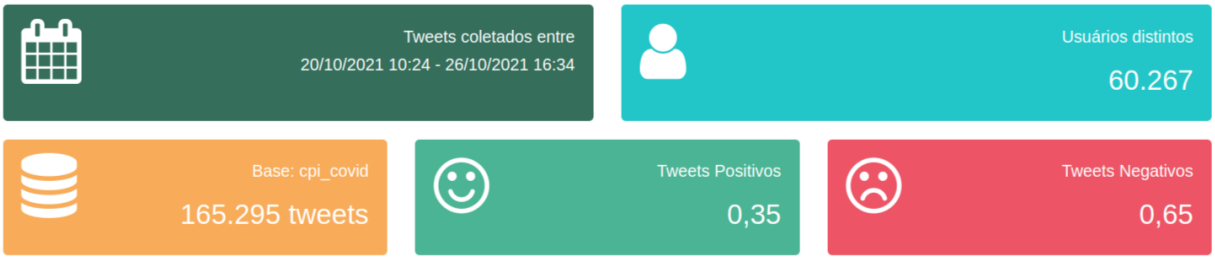


Figura 10 – Consolidados dos dados coletados no período.

Esta seção destina-se a apresentar a sumarização/visualização dos dados referentes a toda a base de *tweets* coletados. Essas informações, dentro da plataforma, são apresen-

tadas na *dashboard* principal. A Figura 10 exibe esse resumo no qual é possível verificar, além dos dados de coleta apresentados na seção anterior, a análise de sentimento consolidada se mostrou no geral muito mais negativa nas postagens coletadas. Essa informação é corroborada pela funcionalidade de nuvem de palavras exibida na Figura 11 (visualização na qual os termos mais recorrentes são evidenciados pelo tamanho proporcional à sua ocorrência). Nela, é possível visualizar muitos termos de cunho negativo, como "vagabundo", "indignação", "circo", "vergonha", dentre outros.

Trazendo à tona o contexto do momento em que a coleta foi realizada, esses são resultados coerentes com o sentimento observado no coletivo, que demandava explicações e responsabilização pelos insatisfatório desempenho do país ao lidar com a pandemia do COVID-19 (CHADE, 2021). A Figura 12, por sua vez, utiliza da mesma representação visual para exibir os pares de termos que co-ocorrem, segundo a modelagem de tópicos utilizando a matriz de fatoração negativa (NFM) com TF-IDF. Da mesma forma, a frequência da ocorrência implica no tamanho apresentado na visualização. É possível verificar através da imagem, o reforço da impressão anterior, dada a presença de muitos termos de cunho negativo.

Figura 11 – Nuvem de termos com maior ocorrência

Outra informação condensada na *dashboard* principal consiste na apresentação do mapa de calor, que representa a distribuição dos usuários ao longo do território nacional. Essa funcionalidade é implementada com base nas informações de latitude e longitude presentes nos *tweets* (para os quais os usuários habilitaram o compartilhamento). Como é possível verificar na Figura 13(a), a maior parte dos conteúdos disseminados se concentram

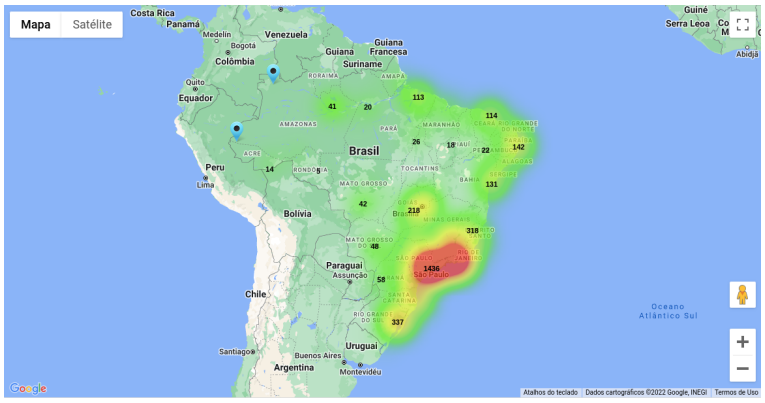
Figura 12 – Nuvem de palavra com a modelagem de tópicos utilizando NMF com TF-IDF

na região sudeste do território nacional, cenário comum e esperado dada a distribuição populacional dessa região. Na funcionalidade é possível ainda, ao ampliar a visão do mapa, ter acesso aos usuários e suas informações de forma individual (como aparece na imagem para usuários na borda da região norte).

Como a disponibilidade dessa informação depende do compartilhamento pelo usuário da mesma, a plataforma pode ter uma limitação em sua amostra a depender do cenário de coleta. Como um *fallback* para situações onde não há informações suficientes de geolocalização dos usuários, e também como visualização adicional a ser agregada, foi incorporado junto ao mapa de calor o mapa de tendências do Google Trends[1] (exibido na figura 13(b), que traz informações de interesse sobre o tópico no mesmo período da coleta. Na figura é possível observar que, diferentemente do mapa de calor dos usuários, os estados que mais se destacaram foram Alagoas e Rondônia. Dois tópicos de interesse respectivos podem explicar esse alto interesse no período: a investigação com relação a um possível "laranja" no contrato com a Covaxin [2] e a atuação com papel de destaque do senador Marcos Rogério, de Rondônia[3].

O componente apresentado na Figura 13, obtido fazendo um processamento nos *links* encontrados nos *tweets* coletados, apresenta as principais fontes de conteúdo utilizadas pelos usuários da base. Ao consolidar os prefixos/radicais desses endereços, a *dashboard* fornece um resumo das plataformas que foram referências no evento da coleta. Um contra-

1 https://trends.google.com/trends/
2 https://g1.globo.com/politica/cpi-da-covid/noticia/2021/08/25/cpi-investiga-se-vendedor-de-alagoas-foi-usado-como-laranja-de-empresa-fiadora-de-contrato-da-covaxin.ghtml
3 https://www.cnnbrasil.com.br/tudo-sobre/marcos-rogerio/



(a) Mapa de Calor dos Usuários

(b) Mapa de interesse no período pelo Google Trends

ponto dessa funcionalidade, que pode ser observado na listagem da imagem, é a ocorrência de algumas plataformas que fornecem pouca informação, principalmente encurtadores de *link*, que fazem o redirecionamento para um novo endereço, "mascarando"essa informação. Uma possibilidade para contornar esse empecilho é implementar a descoberta do endereço de redirecionamento e fazer a substituição.

Outra possibilidade a ser considerada como avanço na investigação das URL's coletadas seria utilizar essas informações como atributo do usuário, de forma que seria possível, por exemplo, utilizá-lo para caracterização e detecção de comunidades de usuários (BARBOSA, 2022). Nos resultados do trabalho, utilizando essa metodologia (aplicado também em um contexto político), verificou-se uma coerência nas comunidades identificadas de acordo com o papel exercido no contexto da coleta.

Principais Fontes (nº de links)	
youtu.be	135
noticias.uol.com.br	125
www.oantagonista.com	91
www.youtube.com	85
revistaforum.com.br	84
www.google.com	61
www.brasil247.com	57
g1.globo.com	53
exame.com	47
www.poder360.com.br	43
bit.ly	42

Figura 13 – Principais fontes de conteúdo

5.2.2 Análise de Sentimentos via classificador (previamente treinado)

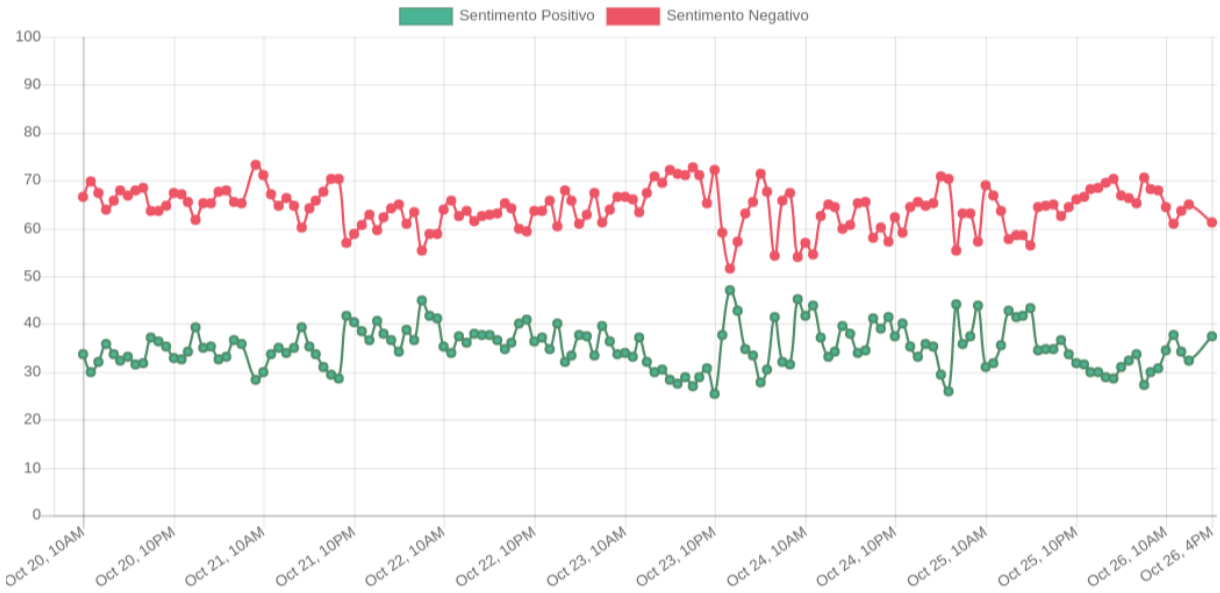


Figura 14 – Análise de Sentimento ao longo do período do evento

Apesar da sumarização da *Dashboard* apresentada anteriormente constar os valores finais da análise de sentimento na base, em muitos cenários é importante entender também sua variação temporal, já que eventos relevantes podem ser camuflados com o resumo dessa informação. Ignorar essa série temporal seria como por exemplo, em um cenário de análise das eleições, analisar apenas o resultado final, sem considerar o comportamento da apuração ao longo do tempo. A Figura 14 exibe, portanto a variação do sentimento (positivo/negativo) na base durante toda a janela de coleta, e sua visualização está presente também na página principal da aplicação.

É possível observar através da imagem que, durante todo o período, os conteúdos foram classificados com maiores índices negativos. Chama atenção, entretanto, dentre as oscilações, a variação apresentada no início do dia 24 de outubro, na qual os indicativos quase se encontram, mas logo após tomam direções opostas. A observação desse comportamento permite relacionar, por exemplo, a notícias (através da definição de intervalos de datas na pesquisa) ou mesmo os próprios *tweets* coletados (exportando os dados também para o intervalo de interesse), que indiquem possíveis causas dessa mudança da polaridade dos sentimentos.

Para uma análise de qualidade do modelo obtido, durante a atualização do modelo, é calculada também a perda (*loss*) do algoritmo de aprendizado em *deep learning* utilizado pela biblioteca, índice que deve ser minimizado. A Figura 15 ilustra essa métrica ao longo do treinamento. É possível verificar que, apesar da rápida queda, o valor já se inicia em uma grandeza muito baixa (1ê-5), isso devido à própria característica da base, que é

toda populada com *emojis* que ajudam a rotular com precisão o sentimento do tweet. Entretanto, aplicando a outros cenários práticos, pode ser identificado um *overfitting* do modelo (o que é esperado e conhecido como limitação), já que como identificado no trabalho relacionado de (ARAÚJO, 2013), uma amostra muito baixa (menos de 10%) dos conteúdos orgânicos contam com esses elementos identificando explicitamente a emoção. Ainda assim, como explicitado também pelo autores, essa técnica vem sendo amplamente utilizada de forma combinada com outros métodos.

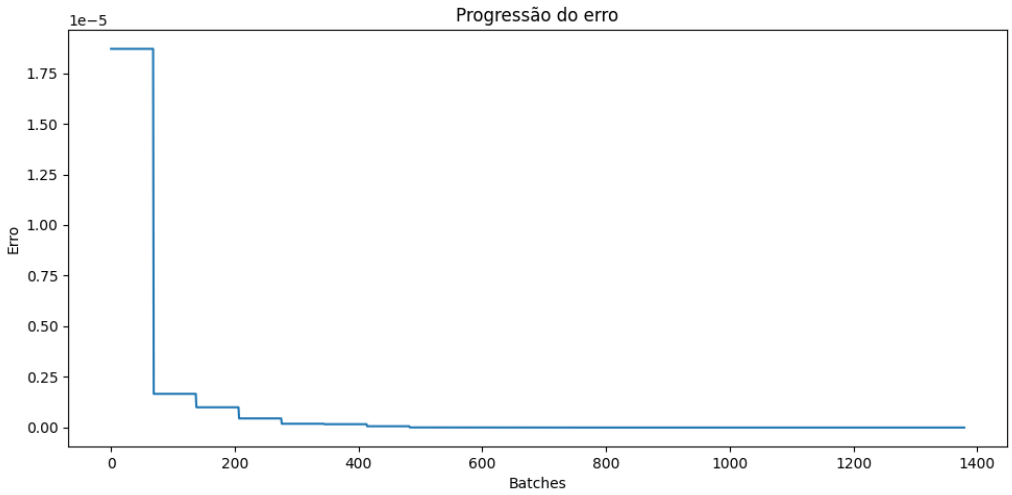


Figura 15 – Perda (*loss*) do algoritmo de rede neural ao longo do treinamento

Ainda a fim de avaliar a performance do modelo dentro da base (onde os rótulos são conhecidos e a eficácia pode ser calculada), foi utilizado o método de *cross-validation*, utilizando partições 70/30 (70% para treinamento e 30% para testes). A Figura 16 exibe a matriz de confusão desse teste através de um mapa de cores. Através dele é possível calcular as métricas de assertividade do modelo e entender como foram as estimativas (0 indicando o sentimento negativo e 1 como sentimento positivo). A diagonal principal indica as previsões corretas dentro do contexto de teste, e consequentemente sua razão sobre o total de amostras indica a acurácia do modelo. Para a base utilizada, a acurácia obtida foi de aproximadamente 70%.

5.2.3 Séries Temporais dos Principais Tweets

Ainda visando detalhar o comportamento da base ao longo do período de coleta, em uma visualização a parte na plataforma, os principais *tweets* podem ser inspecionados a fim de se observar o crescimento no número de usuários que o compartilharam. Na caixa de seleção, exibida ao topo, uma lista com os *tweets* ordenados pelo número de compartilhamentos permite alternar entre os tópicos. Assim também como no caso da série

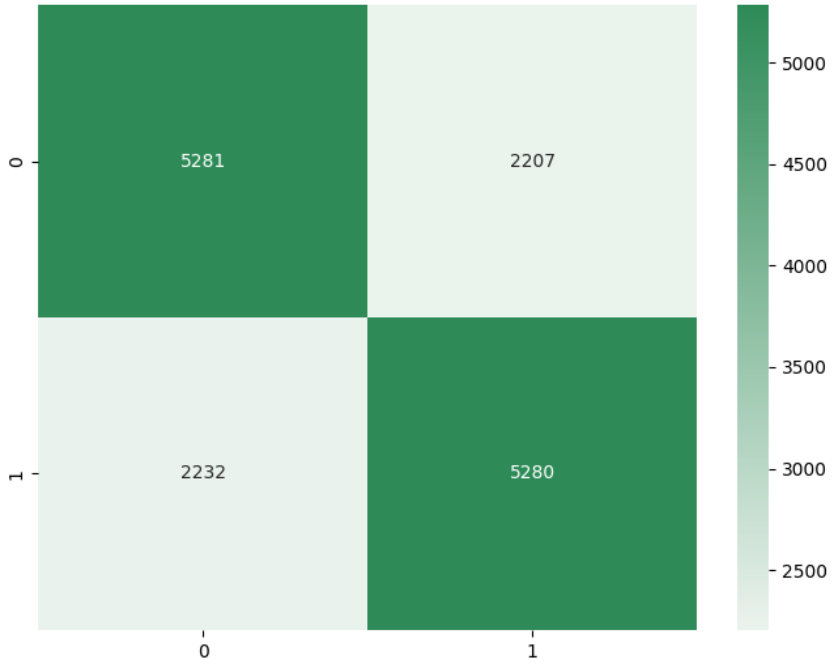


Figura 16 – Matriz de confusão do teste do modelo obtido

temporal da análise de sentimento, essa visualização permite a inspeção para identificação de eventuais correlações e de causas relacionados a eventos dentro do contexto da coleta.

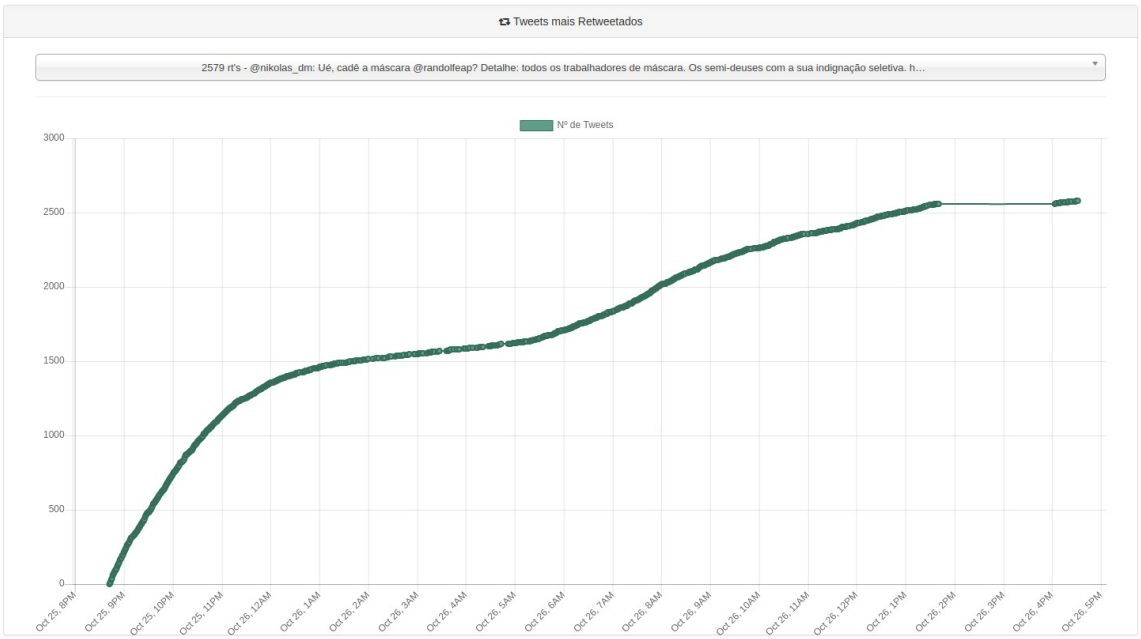


Figura 17 – Exemplo de gráfico com a série temporal do *tweet* de @nikolas_dm (mais compartilhado na base)

A figura 17 exibe o *tweet* mais compartilhado na base, de autoria do usuário Nikolas

Ferreira (@nikolas_dm, na época vereador de Belo Horizonte), que foi publicado já no fim da janela de coleta (25 de outubro), questionando o vice-presidente da CPI, Randolfe Rodrigues (@randolfeap) sobre o uso de máscara. No gráfico, é possível observar uma crescente inicial nos compartilhamentos, logo após estabilizando, seguido de uma nova crescente. Um comportamento parecido pode ser observado no *tweet* do escritor Bernardo Küster (@bernardokuster2), exibido na figura 18 (incluindo também um hiato semelhante entre a última crescente e os compartilhamentos finais), possuindo entretanto uma janela de tempo totalmente distinta do *tweet* anterior, ocorrendo ao início da coleta.

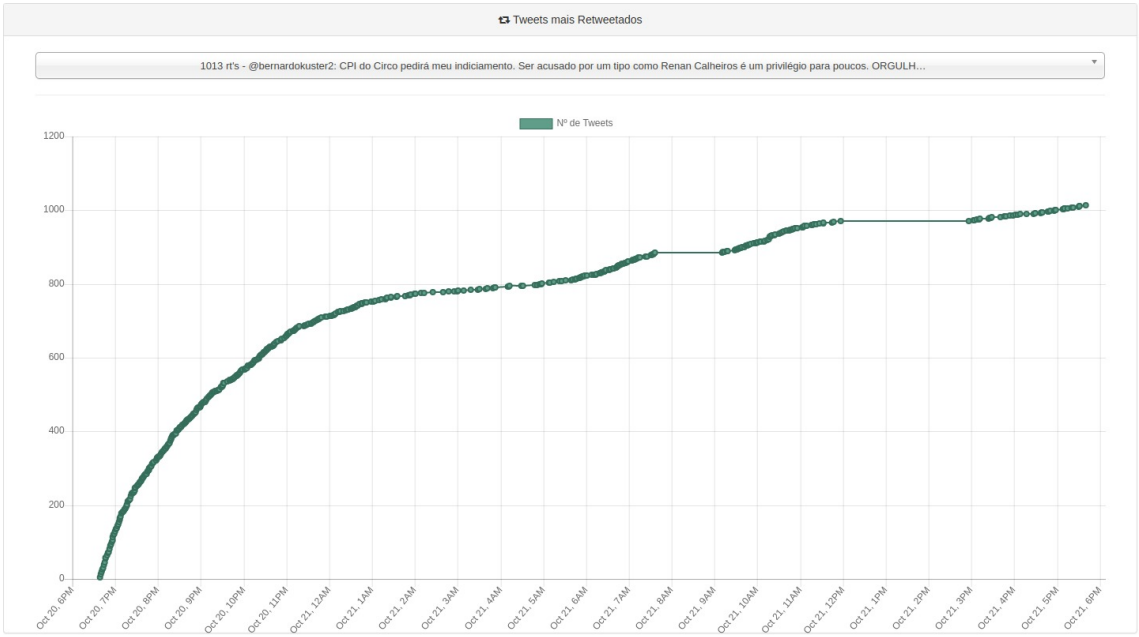


Figura 18 – Exemplo de gráfico com a série temporal do *tweet* de @bernardokuster2

5.2.4 Usuários Mais Centrais

A apresentação dos usuários que se comportam como nós mais centrais da rede se divide em 2 seções, como mostra a figura 19. Enquanto a primeira considera uma ordenação baseada somente no número absoluto de seguidores, a segunda tem como base a modelagem direcionada da rede que reflete nas interações entre os usuários. A imagem evidencia uma grande alternância entre os usuários segundo os critérios analisados. Excetua-se, no entanto, as métricas de grau de entrada e PageRank na rede de menções, que apesar de não apresentarem a mesma ordenação, selecionaram os mesmos usuários como mais centrais. Isso é esperado dado que, apesar da ponderação relativa aos usuários vizinhos, o algoritmo de PageRank ainda tem como critério-base o número de ligações.

Do ponto de vista de uma análise mais qualitativa dos resultados obtidos, é possível verificar que esses usuários que tiveram um maior número de interações (e portanto de arestas na rede modelada) são atores relevantes dentro do contexto (no caso, políticos que participaram diretamente da CPI) e portanto tiveram maior notoriedade e influência.

A métrica de intermediação (*betweenness*), como esperado devido a sua característica, identifica os usuários que funcionam como ”pontes“ nessa rede (dentro da rede social do Twitter, por exemplo, são usuários que comentam, citam e interagem com os atores principais citados anteriormente, permitindo mais visibilidade nas publicações). O número absoluto, por sua vez, dá uma medida de influência mais generalista, que não reflete necessariamente dentro do contexto, mas seleciona usuários de atuação mais abrangente e que eventualmente tiveram alguma interação dentro do cenário da coleta.

Para alguns desses usuários, é válido trazer o contexto e destacar sua atuação durante a CPI do COVID. O senador Marcos Rogério (PL), por exemplo, citado anteriormente no mapa de calor como possível causa do alto interesse no estado de Rondônia, figura entre os usuários mais centrais nas métricas de grau de entrada e PageRank. Também têm lugar de destaca o presidente e vice-presidente do relatório da CPI, os senadores Omar Aziz (PSD-AM) e Randolfe Rodrigues (Rede-AP). O senador Renan Calheiros (MDB-AL), por sua vez, foi responsável pela leitura do relatório. Nos usuários que fazem o papel de intermediação na rede, o aparecimento do ator José de Abreu não é grande surpresa, dado que o mesmo tem se engajado constantemente em assuntos políticos.

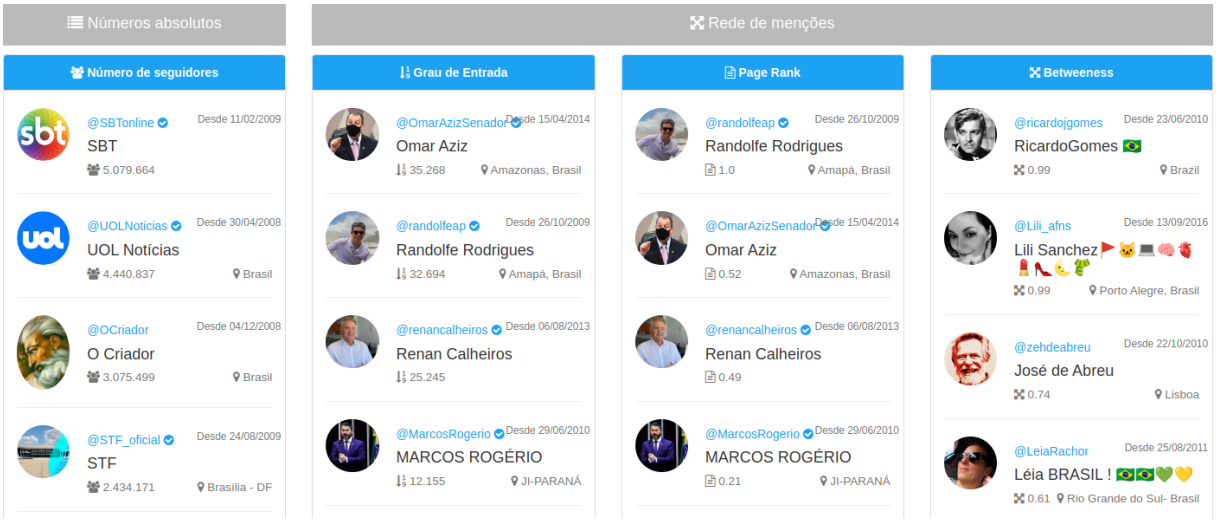


Figura 19 – Usuários mais centrais/influentes segundo cada métrica aplicada

5.2.5 Potenciais Perfis Não-Autênticos

Conforme identificado em trabalhos semelhantes na literatura, a identificação de usuários com comportamento automatizado (robôs) é um desafio que esbarra na incerteza quanto ao rótulo determinado, uma vez que mesmo uma análise humana pode não ser conclusiva ou até mesmo errônea. Conhecida essa limitação, foi utilizado um limiar rigoroso para nota de corte no *score* que determina quais usuários serão exibidos. Seguindo

este critério, verificou-se que um total de 4.772 usuários (cerca de 8% de todos os usuários da base) foram identificados como potenciais perfis não-autênticos.

Na Figura 20 é possível ver uma amostra desses perfis através de uma *datatable* que permite a visualização paginada, bem como a busca e ordenação pelos atributos. Em uma inspeção para análise mais qualitativa, verifica-se um padrão comum dentre esses usuários: foram criados recentemente, possuem nomes de usuários que indicam uma criação automática (nome + número aleatório ao final), quase nenhum seguidor, e um comportamento de apenas re-compartilhar outros conteúdos (nenhum conteúdo autoral). Em todas as funcionalidades implementadas, esses usuários não foram desconsiderados. Entretanto, para fins de confiabilidade, a depender da aplicação, pode ser feito um pré-processamento na base para remoção do conteúdo publicado por esses autores.

Possíveis Perfis Falsos

Pesquisar

Score	Usuário	Nome	Seguidores	Seguindo	Status	Listado em	Fonte
-2	@Mari195377	Mari1953	0	2	41	0	Twitter for Android
-2	@Ivonete79394871	Ivonete	0	16	2	0	Twitter for Android
-2	@AlineSo93494871	Aline Soares	0	2	1	0	Twitter for Android
-2	@CABRALC09970383	CABRAL, Cabral	0	8	9	0	Twitter for Android
-2	@Carla72045613	Carla	4	326	31	0	Twitter for Android
-2	@Francis88816573	Francismar	0	31	2	0	Twitter for Android
-2	@Mary154488061	Mary	0	76	8	0	Twitter for Android
-2	@Cem62703847	Cem	6	316	97	0	Twitter Web App

Figura 20 – *Datatable* listando alguns dos potenciais perfis falsos identificados na base

5.2.6 Modelagem de Tópicos (LDA)

Os tópicos obtidos pelo LDA e seus termos respectivos plotados na figura 21, revelam, no geral, como os termos de descontentamento identificados nas nuvem de palavras melhor se agrupam e se relacionam. O primeiro tópico, por exemplo, traz como foco termos de indignação a respeito de aglomeração e uso (ou ausência) de máscara e envolvem o presidente da CPI, o senador Omar Aziz, citando anteriormente. O segundo tópico, por sua vez, aparenta a relação da vacina (ou, no contexto, sua falta/atraso) com o número de mortos na pandemia. Já o tópico 5 projeta o impacto desse cenário nas eleições que se seguiriam (o que efetivamente verificou-se, através do resultado, ter um alto custo nos índices de satisfação da população com o governo atual). Cada um dos demais tópicos consegue trazer ainda, de forma visualmente intuitiva, os termos presentes em cada grupo, bem como sua importância no assunto em questão.

Enquanto a nuvem de palavras é um recurso visual de fácil implementação/processamento, ela tende a obscurecer os tópicos subjacentes da discussão e permitem que

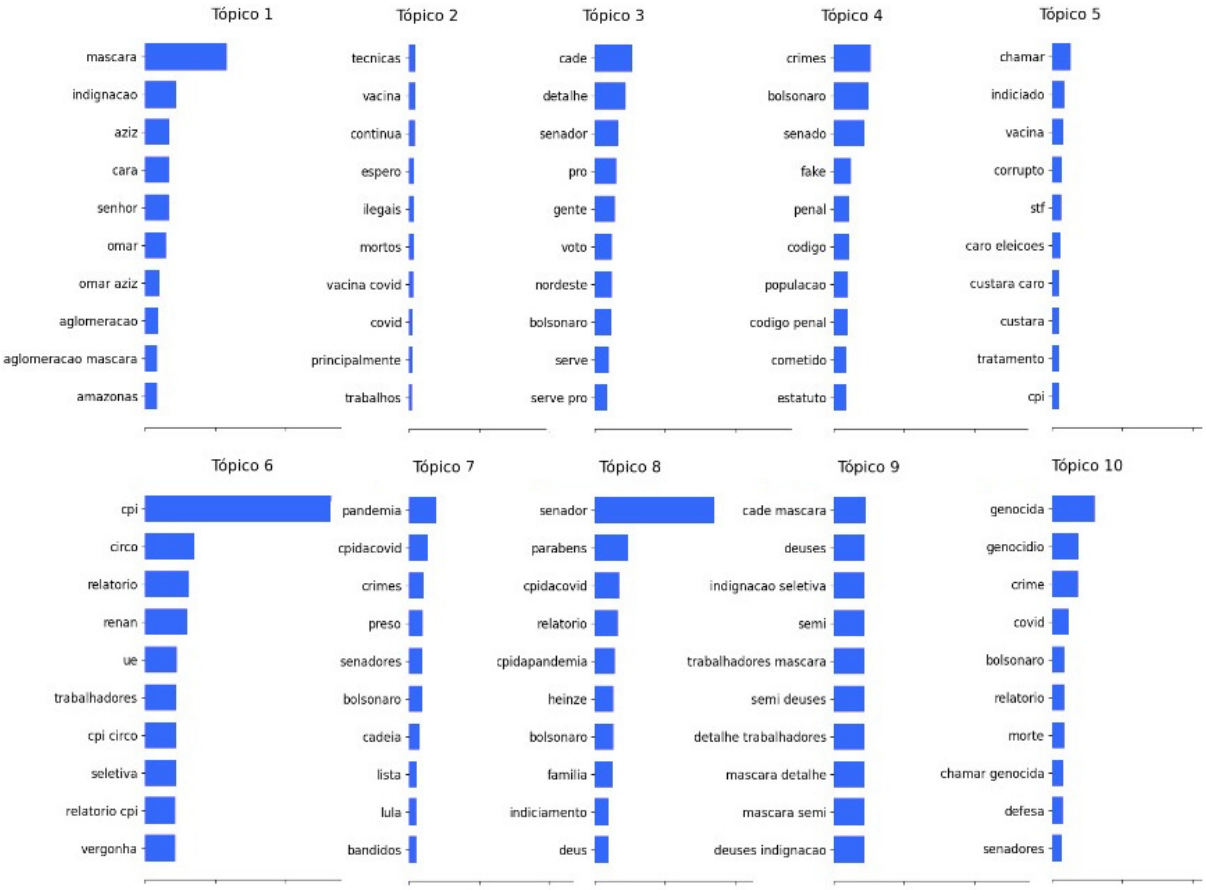


Figura 21 – Tópicos e termos respectivos obtidos utilizando o LDA

apenas as palavras usadas com mais frequência apareçam. A modelagem de tópicos, por outro lado, é focada em descobrir tópicos implícitos em uma coleção de dados, de modo a descobrir a estrutura semântica mais oculta no conteúdo. Quando implementada corretamente, a modelagem de tópicos pode ser uma ferramenta muito poderosa na análise de mídia social. A modelagem de tópicos permite o particionamento de conversas, que podem ter referenciado a mesma *hashtag*(#) ou palavras-chave, em tópicos distintos ou conversas focadas em um conjunto distinto de tópicos. Uma vez que os tópicos são abstratos e derivados de máquina, eles são apresentados como um conjunto de palavras mais usadas que são representativas do tópico e um *score* (apresentado na visualização pelo tamanho da barra ao lado do termo) que representa a importância da palavra no tópico.

5.3 Caracterização e Análise de Comunidades

Utilizando os parâmetros descritos na metodologia (que são os valores padrão e sugeridos pela literatura), observou-se que o *backbone* extraído retorna pouco mais de 10% dos usuários principais enquanto o número de arestas foi reduzido para menos de 4% da rede original, reduzindo significativamente a complexidade e custo computacional a serem

trabalhados. Os parâmetros descritos podem ser observados nos *inputs* da Figura 22, que também inclui o cabeçalho com alguns dos principais números descritivos das comunidades obtidas. Outra informação importante presente nos cabeçalhos é a medida de qualidade das comunidades: a modularidade. O valor 0,62 se mostra satisfatório uma vez que o intervalo possível é [-0.5,1] e valores mais próximos de 1 indicam uma comunidade forte.

Logo abaixo ao cabeçalho, um *input* de seleção lista as comunidades obtidas, ordenadas pelo maior número de usuários presentes. Ao selecionar uma comunidade, a *dashboard* apresentada reflete a visualização dos dados dentro do escopo da rede escolhida. Para análise de resultados, foram consideradas as duas maiores sub-redes obtidas, uma vez que foi observado que ambas representavam polaridades distintas do âmbito político.

Figura 22 – Dados gerais das comunidades obtidas

As Figuras 23 e 24, que resumem alguns dados de cada uma das respectivas comunidades, consolidam todos os dados coletados ao longo do tempo. Para a análise de sentimento, por exemplo, o valor final consolidado foi bem próximo para ambas, refletindo um sentimento negativo no geral. Entretanto, esse comportamento que variou ao longo do tempo, pode ser melhor visualizado nas Figuras 25 e 26. É válido ressaltar também que, ainda que a análise de sentimento busque polarizar o teor do discurso, é necessário interpretar também o objeto sobre o qual o discurso se refere. Em um contexto político polarizado, por exemplo, ambos os lados podem apresentar discursos de descontentamento e teor negativo, direcionados à polaridade oposta.

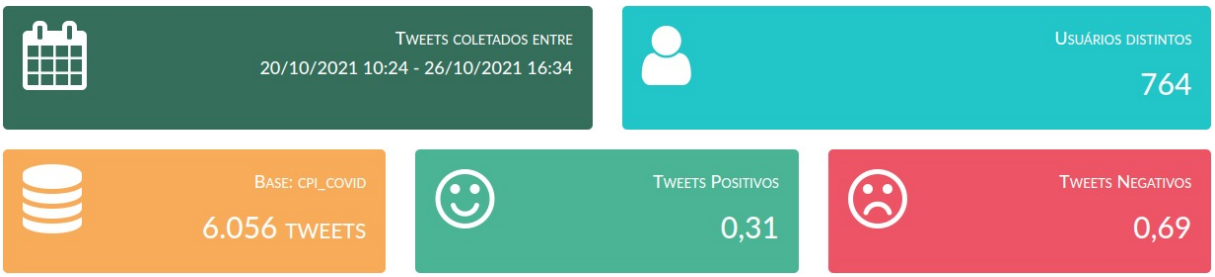


Figura 23 – Sumarização da maior comunidade da rede

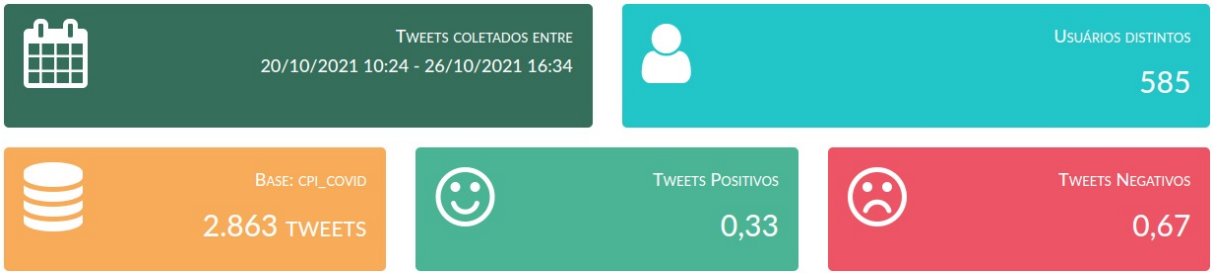


Figura 24 – Sumarização da segunda maior comunidade da rede

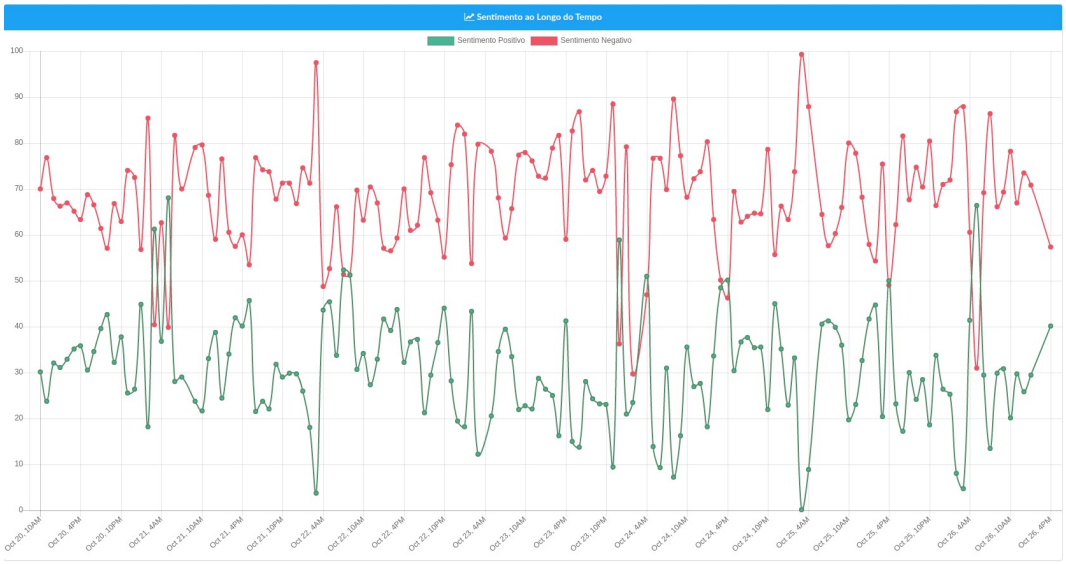


Figura 25 – Variação do sentimento ao longo do tempo na maior comunidade

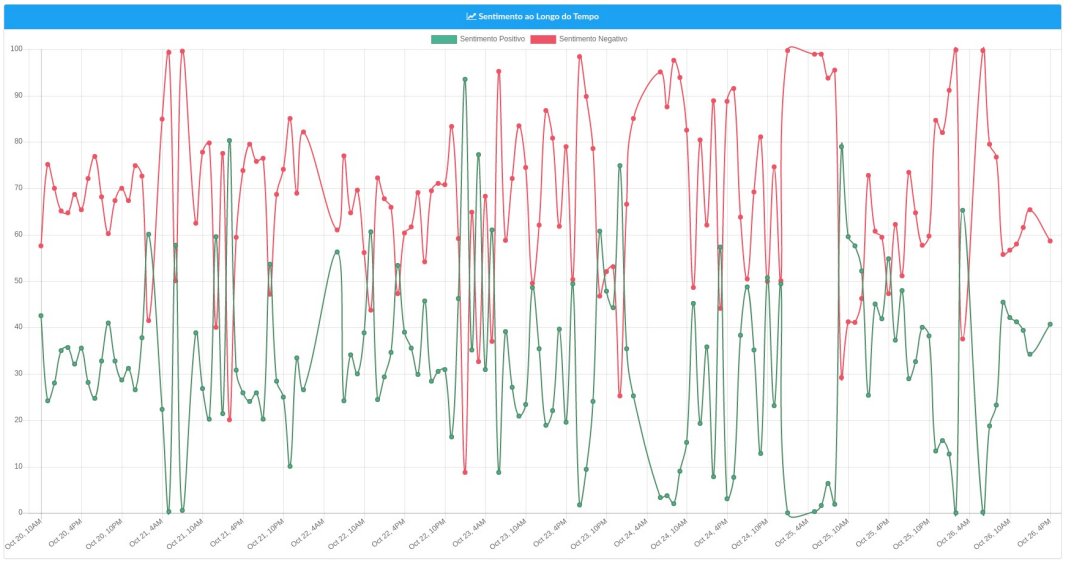
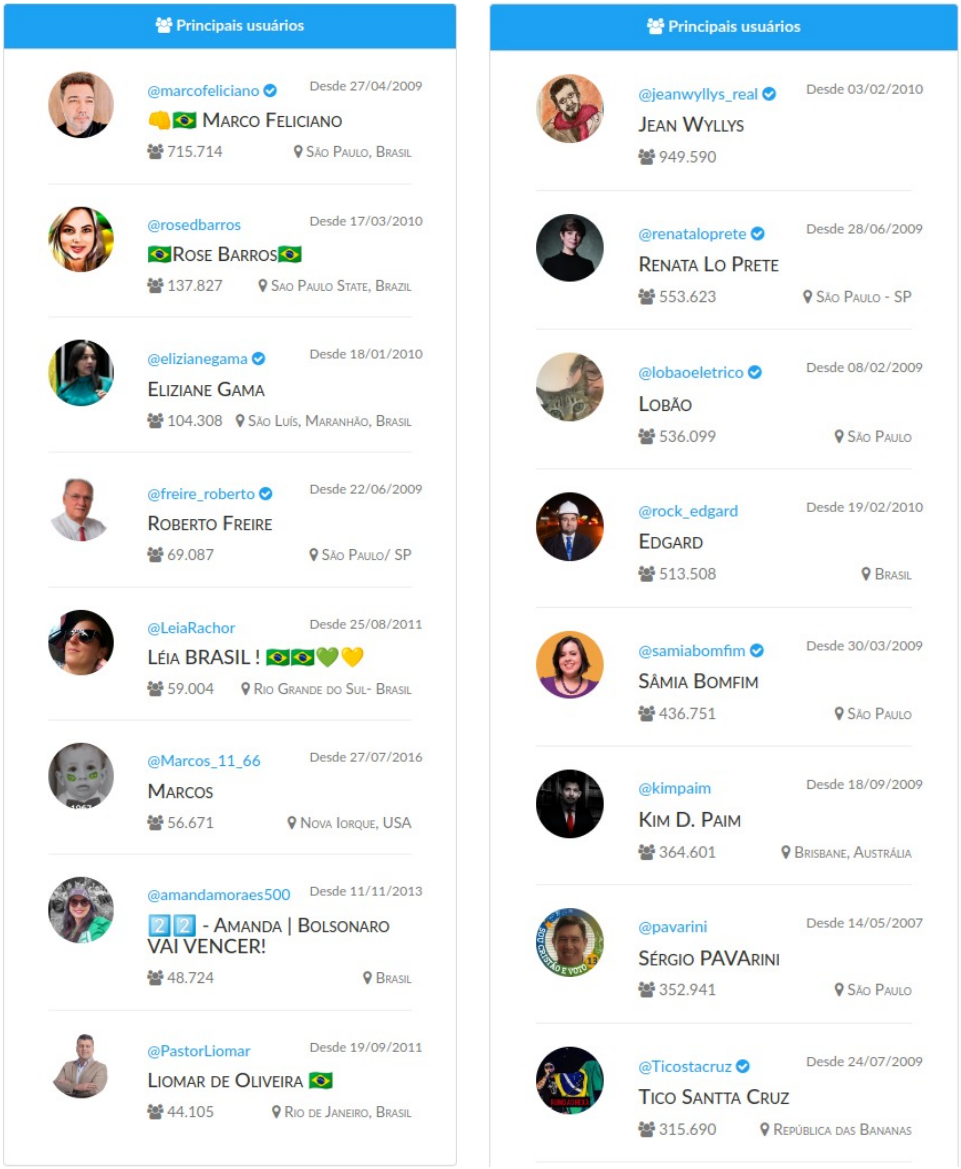


Figura 26 – Variação do sentimento ao longo do tempo na segunda maior comunidade

A informação apresentada que permite entender melhor a caracterização da rede, no contexto de estudo, foram os principais atores/usuários (pelo critério do número de

seguidores), apresentados na Figura 27. É possível verificar, através de uma análise qualitativa, que os usuários da maior rede, pertencem, no geral, a um posicionamento político de direita (como Marco Feliciano, Rose Barros e Pastor Liomar de Oliveira). A segunda maior comunidade, por sua vez, apresenta o posicionamento político oposto, de esquerda (Jean Wyllys, Lobão e Tico Santa Cruz são alguns atores da rede ativos em seu posicionamento). Essa verificação traz uma informação importante para análise das informações apresentadas dentro do escopo de cada comunidade.



(a) Maior comunidade da rede (b) Segunda maior comunidade da rede

Figura 27 – Principais usuários segundo critério de número de seguidores

Observando a polaridade identificada, através das nuvens de palavras apresentadas nas Figuras 28 e 29, é possível visualizar uma síntese dos principais termos de cada comunidade. Para a maior comunidade (de posicionamento predominante de direita), a maior parte dos termos expressam pouco sentimento/emoção e se referem mais a atores e objetos

relacionados ao evento (como ”relatório”,”senador”,”presidente”,”bolsonaro”,”senado”, dentre outros), não evidenciando tão claramente o teor das publicações quanto a segunda comunidade. Esta, por sua vez (de posicionamento predominantemente de esquerda), apresenta alguns termos que indicam claro descontentamento, como por exemplo: ”genocida”, ”vagabundo”, ”crimes”, ”circo”.

Figura 28 – Nuvem de palavras na maior comunidade

5.4 Conclusão sobre os resultados

Foi possível verificar, através da análise de resultados no cenário aplicado, que a integração das funcionalidades (bem como a diversidade de ferramentas e visualizações) contribuem para inferências mais assertivas. Cada informação adicional elucida um pouco mais o caso de estudo com novas visões implícitas ao contexto e traz maior credibilidade às conclusões que, sem essas múltiplas fontes de dados corroborando a afirmação, talvez tivessem seu argumento enfraquecido. A implantação de novas funcionalidades, juntamente com a melhoria das existentes (que possuem muita margem para exploração e otimização), portanto, é altamente desejável para aprimorar ainda mais as análises auxiliadas pela plataforma.

Figura 29 – Nuvem de palavras na segunda maior comunidade

6 Conclusão

O trabalho aqui desenvolvido, bem como os resultados obtidos, mostraram algumas das possibilidades de aplicação prática da ciência de dados e descoberta de conhecimento aplicadas no contexto de redes sociais. Em contextos similares à proposta do projeto (onde se encontra um grande volume de dados que dificultam sua visualização e interpretação), as funcionalidades e métodos aqui apresentados demonstram possibilidades de como a ferramenta desenvolvida pode apoiar desde o processo da coleta e processamento, até a tomada de decisão. Na eventual utilização efetiva da plataforma, em diálogo com profissionais de outras áreas (que não demandem um alto conhecimento técnico para utilização da aplicação), seria possível identificar melhorias e novas funcionalidades a serem agregadas.

Os resultados demonstrados permitiram visualizar a consolidação e sumarização de informações relevantes no contexto da coleta, bem como observar indicadores importantes, referentes aos termos de interesse da busca. O cruzamento dessas informações possibilitam a extração de conhecimentos subjacentes e obtenção de *insights* para análise do cenário. A plataforma permitiu avaliar também o comportamento da rede coletada sob a perspectiva de formação de comunidades, bem como a identificação de aspectos relevantes no entendimento e caracterização de sua formação e seus principais atores.

A integração entre interesse em pesquisa acadêmica e a aplicação prática em desenvolvimento de *software* aqui explorados demonstram também como a ciência de dados é uma área de interesse compartilhado tanto pela academia quanto pelo mercado de trabalho. Essa integração é somente possível graças às inúmeras áreas de conhecimento que a compõem (mineração e visualização de dados, análise estatística, aprendizado de máquina, dentre outras). Em consequência dessa ampla aplicabilidade, é uma das áreas com maior crescimento de interesse nos últimos anos.

Para trabalhos futuros, algumas possibilidades se apresentam no formato atual da aplicação. Para as funcionalidades já desenvolvidas, algumas melhorias possibilitariam uma melhor análise das informações obtidas. Por exemplo, na visualização de domínios dos conteúdos, o tratamento de URL's que utilizem encurtadores de *links* possibilitaria um melhor agrupamento dessas informações. Um passo a mais na obtenção de informações via URL seria obter, para canais de conteúdo (como Youtube, Tiktok, dentre outros) não só o domínio, mas também o canal ou usuários referentes.

Para a detecção de comunidades, uma visualização otimizada para redes de larga escala seria uma boa representação visual a ser agregada. É possível ainda explorar pontos de melhoria para os algoritmos de aprendizado/predição utilizados, uma vez que aqueles utilizados não são o estado da arte em sua proposta. Por fim, dada a característica modular

da aplicação que permite facilmente a integração de novas funcionalidades, seria possível explorar a extração de outras visualizações, ou ainda incluir outras redes sociais e fontes de informação para alimentar a base utilizada.

Referências

SILVA, V. B. da. Marketing digital como ferramenta estratégica e as oportunidades nas redes sociais: Digital marketing as a strategic tool and opportunities in social networks. *e3-Revista de Economia, Empresas e Empreendedores na CPLP*, v. 2, n. 1, p. 42–61, 2016. Citado na página 11.

ZHAO, S.; ZHONG, L.; WICKRAMASURIYA, J.; VASUDEVAN, V. Human as real-time sensors of social and physical events: A case study of twitter and sports games. *arXiv preprint arXiv:1106.4300*, 2011. Citado na página 11.

PARK, C. S. Does twitter motivate involvement in politics? tweeting, opinion leadership, and political engagement. *Computers in human behavior*, Elsevier, v. 29, n. 4, p. 1641–1648, 2013. Citado na página 11.

FRANÇA, T. C.; FARIA, F. F. d.; RANGEL, F. M.; FARIAS, C. M. d.; OLIVEIRA, J. Big social data: Princípios sobre coleta, tratamento e análise de dados sociais. *XXIX Simpósio Brasileiro de Banco de Dados–SBBD*, v. 14, 2014. Citado na página 11.

RAPPIE, C. Hootsuite for academia? how to increase the visibility, downloads and impact of publications using kudos. *Impact of Social Sciences Blog*, Blog post from London School of Economics & Political Science, 2016. Citado na página 11.

PEZZINI, A. Mineração de textos: conceito, processo e aplicações. *REVISTA BRASILEIRA DE CONTABILIDADE E GESTÃO*, v. 5, n. 8, p. 58–61, 2017. Citado na página 12.

TOWARDS AI. *Text Mining in Python: Steps and Examples*. 2019. https://towardsai.net/p/data-mining/text-mining-in-python-steps-and-examples-78b3f8fd913b. Acessado em 07/09/2022. Citado na página 14.

LEE, D. D.; SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature*, Nature Publishing Group, v. 401, n. 6755, p. 788–791, 1999. Citado 2 vezes nas páginas 14 e 16.

STECANELLA, B. *Understanding TF-ID: A Simple Introduction*. 2019. https://monkeylearn.com/blog/what-is-tf-idf/. Acessado em 22/12/2021. Citado na página 14.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, v. 3, n. Jan, p. 993–1022, 2003. Citado 2 vezes nas páginas 14 e 17.

CHOUBEY, V. *Topic Modelling Using NMF*. 2020. https://medium.com/voice-tech-podcast/topic-modelling-using-nmf-2f510d962b6e. Acessado em 17/08/2022. Citado na página 16.

TERRY-JACK, M. *NLP: Pretrained Named Entity Recognition (NER)*. 2019. https://medium.com%2F\spacefactor\@m{}medium.com/\spacefactor\@m{}b.terryjack/nlp-pretrained-named-entity-recognition-7caa5cd28d7b. Acessado em 04/05/2022. Citado na página 17.

HAMADA, L.; NETO, N. Desambiguação de homógrafos-heterófonos por aprendizado de máquina em português brasileiro. In: *Anais do X Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*. Porto Alegre, RS, Brasil: SBC, 2015. p. 181–190. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/stil/article/view-/397>. Citado na página 20.

WEST, D. B. *Introduction to graph theory*. [S.l.]: Prentice hall Upper Saddle River, 2001. Citado na página 20.

ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, APS, v. 74, n. 1, p. 47, 2002. Citado na página 20.

SAXENA, A.; IYENGAR, S. Centrality measures in complex networks: A survey. *CoRR*, abs/2011.07190, 2020. Disponível em: <https://arxiv.org/abs/2011.0719>. Citado na página 20.

SERRANO M. BOGUÑÁ, M. V. A. Extracting the multiscale backbone of complex weighted networks. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 106, n. 16, p. 6483–6488, 2009. Citado 3 vezes nas páginas 21, 24 e 36.

BLONDEL V. GUILLAUME, J. L. R. L. E. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, IOP Publishing, v. 2008, n. 10, p. P10008, 2008. Citado 2 vezes nas páginas 21 e 26.

FREEMAN, L. C. A set of measures of centrality based on betweenness. *Sociometry*, JSTOR, p. 35–41, 1977. Citado na página 21.

KOURTELLIS, N.; ALAHAKOON, T.; SIMHA, R.; IAMNITCHI, A.; TRIPATHI, R. Identifying high betweenness centrality nodes in large social networks. *Social Network Analysis and Mining*, Springer, v. 3, n. 4, p. 899–914, 2013. Citado na página 22.

PAGE, L.; BRIN, S.; MOTWANI, R.; WINOGRAD, T. *The PageRank citation ranking: Bringing order to the web*. [S.l.], 1999. Citado na página 23.

GUPTA, P.; GOEL, A.; LIN, J.; SHARMA, A.; WANG, D.; ZADEH, R. Wtf: The who to follow service at twitter. In: *Proceedings of the 22nd international conference on World Wide Web*. [S.l.: s.n.], 2013. p. 505–514. Citado na página 23.

JIANG, B. Ranking spaces for predicting human movement in an urban environment. *International Journal of Geographical Information Science*, Taylor & Francis, v. 23, n. 7, p. 823–837, 2009. Citado na página 23.

BRÖHL, T.; LEHNERTZ, K. Centrality-based identification of important edges in complex networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, AIP Publishing LLC, v. 29, n. 3, p. 033115, 2019. Citado na página 24.

MASLOV, S.; SNEPPEN, K. Specificity and stability in topology of protein networks. *Science*, American Association for the Advancement of Science, v. 296, n. 5569, p. 910–913, 2002. Citado na página 25.

LU, Z.; WAHLSTRÖM, J.; NEHORAI, A. Community detection in complex networks via clique conductance. *Scientific reports*, Nature Publishing Group, v. 8, n. 1, p. 1–16, 2018. Citado na página 26.

CLAUSET, A.; NEWMAN, M. E.; MOORE, C. Finding community structure in very large networks. *Physical review E*, APS, v. 70, n. 6, p. 066111, 2004. Citado na página 27.

SHAO, C.; CIAMPAGLIA, G. L.; VAROL, O.; FLAMMINI, A.; MENCZER, F. The spread of fake news by social bots. *arXiv preprint arXiv:1707.07592*, v. 96, p. 104, 2017. Citado na página 28.

BOVET, A.; MAKSE, H. A. Influence of fake news in twitter during the 2016 us presidential election. *Nature communications*, Nature Publishing Group, v. 10, n. 1, p. 1–14, 2019. Citado na página 28.

KNIGHT, W. *Why It's So Hard to Count Twitter Bots*. 2022. https://www.wired.com/story/twitter-musk-bots/. Acessado em 02/07/2022. Citado na página 29.

CRESCI, S.; PIETRO, R. D.; PETROCCHI, M.; SPOGNARDI, A.; TESCONI, M. Fame for sale: Efficient detection of fake twitter followers. *Decision Support Systems*, Elsevier, v. 80, p. 56–71, 2015. Citado 2 vezes nas páginas 29 e 49.

OLIVEIRA, N. R. de; PISA, P. S.; COSTA, B.; LOPEZ, M. A.; MORAES, I. M.; MATTOS, D. M. Processamento de linguagem natural para identificação de notícias falsas em redes sociais: Ferramentas, tendências e desafios. *Sociedade Brasileira de Computação*, 2020. Citado na página 30.

NASCIMENTO, P. C. Dicionário de polaridades para apoio a análise de sentimento. *COPRE UFRJ*, 2014. Citado na página 30.

LAMPLE, G.; BALLESTEROS, M.; SUBRAMANIAN, S.; KAWAKAMI, K.; DYER, C. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016. Citado na página 33.

BENEVENUTO, F.; ALMEIDA, J. M.; SILVA, A. S. Explorando redes sociais online: Da coleta e análise de grandes bases de dados às aplicações. *Porto Alegre: Sociedade Brasileira de Computação*, p. 22, 2011. Citado na página 35.

ARAÚJO, M.; GONÇALVES, P.; BENEVENUTO, F.; CHA, M. Métodos para análise de sentimentos no twitter. In: SN. *Proceedings of the 19th Brazilian symposium on Multimedia and the Web (WebMedia'13)*. [S.l.], 2013. p. 19. Citado 2 vezes nas páginas 35 e 61.

PESSANHA, G. R. G.; FIDELIS, T. O.; FREIRE, C. D.; SOARES, E. A. # fiqueemcasa: Análise de sentimento dos usuários do twitter em relação ao covid19. *HOLOS*, v. 5, p. 1–20, 2020. Citado na página 35.

LIMA, A. C.; CASTRO, L. D. Uso de emoticons para análise de sentimento de tweets. In: . [S.l.: s.n.], 2012. Citado na página 35.

LEITE, M. A. G. L. Um modelo baseado em regras para a detecção de bots no twitter. UFVJM, 2019. Citado na página 36.

BARBOSA, C. M.; FÉLIX, L. G. d. S.; ALVES, A. P. S.; XAVIER, C. R.; VIEIRA, V. da F. Uso de urls para caracterização de comunidades em redes sociais online. In: SBC. *Anais do XI Brazilian Workshop on Social Network Analysis and Mining*. [S.l.], 2022. p. 25–36. Citado 2 vezes nas páginas 36 e 59.

BARBOSA, C. M. G.; FELIX, L. G. d. S.; XAVIER, C. R.; VIEIRA, V. d. F. A framework for the analysis of information propagation in social networks combining complex networks and text mining techniques. In: *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*. [S.l.: s.n.], 2019. p. 401–408. Citado na página 36.

COURSERA. *Popular Programming Languages in 2022*. 2022. https://www.coursera.org/articles/popular-programming-languages. Acessado em 02/09/2022. Citado na página 40.

CHADE, J. Relatório da cpi da pandemia é documento histórico da crise da covid-19 no mundo. El País (Brasil), 2021. Citado na página 57.