Detecção precoce de rumor: avaliação da eficácia do uso de redes neurais de grafo

Douglas Melo dos Santos

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2025

Douglas Melo dos Santos

Detecção precoce de rumor: avaliação da eficácia do uso de redes neurais de grafo

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Fabíola Souza Fernandes Pereira
Coorientador: Elaine Ribeiro de Faria Paiva
Uberlândia
2025

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

S237d
2025

Santos, Douglas Melo dos, 1997-
Detecção precoce de rumor [recurso eletrônico] : avaliação da eficácia do uso de redes neurais de grafo / Douglas Melo dos Santos. - 2025.

Orientadora: Fabíola Souza Fernandes Pereira.
Coorientadora: Elaine Ribeiro de Faria Paiva.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Programa de Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2025.5213
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Pereira, Fabíola Souza Fernandes, 1987-, (Orient.). II. Paiva, Elaine Ribeiro de Faria, 1980-, (Coorient.). III. Universidade Federal de Uberlândia. Programa de Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

André Carlos Francisco
Bibliotecário-Documentalista - CRB-6/3408

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação, 30/2025, PPGCO				
Data:	22 de Agosto de 2025	Hora de início:	09:03	Hora de encerramento:	11:10
Matrícula do Discente:	12312CCP009				
Nome do Discente:	Douglas Melo dos Santos				
Título do Trabalho:	Detecção precoce de rumor: Avaliação da eficácia do uso de redes neurais de grafo				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Ciência de Dados				
Projeto de Pesquisa de vinculação:	----------				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Elaine Ribeiro de Faria Paiva - FACOM/UFU (Coorientadora), Bruno Augusto Nassif Travençolo - FACOM/UFU , Jean Roberto Ponciano - ICMC/ USP e Fabíola Souza Fernandes Pereira- FACOM/UFU, orientador(ra) do(a) candidato(a).

Os examinadores participaram desde as seguintes localidades: Jean Roberto Ponciano - São Carlos/SP e os outros membros da banca da cidade de Uberlândia/MG. O(a) aluno(a) Douglas Melo dos Santos participou de Arcos-MG.

Iniciando os trabalhos o(a) presidente da mesa, Profª. Drª. Fabíola Souza Fernandes Pereira, apresentou a Comissão Examinadora e o candidato(a), agradeceu a presença do público, e concedeu ao(á) Discente a palavra para a exposição do seu trabalho. A duração da apresentação do(a) Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o(a) senhor(a) presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir ao candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato(a):

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação

Agradecimentos

Agradeço a Deus por ter me dado saúde e sabedoria durante esses 2 anos e meio de mestrado, conciliando com o trabalho, mudanças de vida e desafios pessoais.

Agradeço às minhas orientadoras Fabíola e Elaine por terem me dado todo o suporte e direção para concluir a pesquisa. Com elas, evolui muito como pesquisador e profissional. O impacto que vocês tiveram na minha trajetória foi imensurável.

Também sou grato a minha família por me dar o suporte necessário e entender minha ausência nos momentos que tive que priorizar os estudos.

"Consagre ao Senhor tudo o que você faz, e os seus planos serão bem-sucedidos."
(Provérbios 16:3)

Resumo

A área de pesquisa de combate à desinformação nas mídias sociais tem exigido métodos cada vez mais robustos de detecção de rumores e notícias falsas. É nesse contexto que este estudo avalia o uso de Redes Neurais de Grafos (GNN's) para detecção de rumores, comparando-as com modelos tradicionais da literatura com foco no desempenho de detecção precoce. O método proposto inclui o desenvolvimento de um módulo padronizado para comparativo e teste rápido de algoritmos que garante divisões cronológicas dos dados para evitar vazamentos de informação e refletir melhor a propagação de rumores em cenários reais. Esse módulo possibilita o retreinamento e inferência incremental, para computar as atualizações nos atributos temporais da base de treino e também armazenar as principais métricas de classificação, fornecendo metadados sobre a eficácia na detecção precoce. Utilizando o conjunto de dados PHEME, um grafo heterogêneo foi gerado para modelar as interações entre tweets e usuários. Dois macroexperimentos foram executados para testar as duas GNNs, juntamente com os modelos LightGBM, Random Forest e LSTM.Os resultados mostraram que as GNNs alcançam um melhor equilíbrio entre precisão e revocação do que os modelos que processam dados tabulares, demonstrando seu potencial para detecção de desinformação em tempo real, ao capturar de forma eficaz as relações complexas entre *tweets* e usuários. Esta pesquisa contribui para a área de detecção de rumores por meio do aprendizado em grafos, fornecendo técnicas mais adequadas para a concepção de novos algoritmos, através da padronização de procedimentos com foco na detecção precoce.

Palavras-chave: Detecção de Rumor, Redes Neurais de Grafo, Grafos Heterogêneos.

Abstract

The field of research focused on combating misinformation on social media has increasingly demanded robust methods for detecting rumors and fake news. In this context, this study evaluates the use of Graph Neural Networks (GNNs) for rumor detection, comparing them with traditional models from the literature, with an emphasis on early detection performance.The proposed method includes the development of a standardized module for quick algorithm testing and comparison, ensuring chronological data splits to prevent information leakage and better reflect rumor propagation in real-world scenarios. This module enables retraining and incremental inference to account for updates in the temporal features of the training data. It also stores key classification metrics, providing metadata about the effectiveness in early detection.Using the PHEME dataset, a heterogeneous graph was generated to model interactions between tweets and users. Two main experiments were conducted to test two GNNs, along with the LightGBM, Random Forest, and LSTM models.The results showed that GNNs achieved a better balance between precision and recall than models that process tabular data, demonstrating their potential for real-time misinformation detection by effectively capturing the complex relationships between tweets and users. This research contributes to the field of rumor detection through graph-based learning, providing more suitable techniques for designing new algorithms by standardizing procedures focused on early detection.

Keywords: Rumor Detection, Graph Neural Networks, Heterogeneous Graphs .

Lista de ilustrações

Lista de tabelas

Sumário

Capítulo 1

Introdução

Rumores são definidos como a divulgação de uma informação não verificada nas primeiras horas após ser divulgada por algum tipo de veículo de informação (ZUBIAGA et al., 2018). O acesso instantâneo a grandes volumes de publicações (notícias ou postagens) com o uso em massa de redes sociais, mantém a população informada e conectada em um mundo dinâmico. Entretanto, informações falsas ou não verificadas podem se espalhar rapidamente, atingir vastos públicos em pouco tempo e frequentemente têm consequências prejudiciais. Durante eventos críticos, como desastres naturais, crises de saúde ou eleições, os boatos têm o potencial de desinformar o público, incitar pânico e minar a confiança nas autoridades (JAFFAR; MOHADES; ABADY, 2025).

Um estudo feito por (RUBIN, 2010) da área da psicologia mostrou que os seres humanos têm a capacidade de identificar entre 55-58% de rumores, indicando a facilidade com a qual o público pode ser influenciado. Após sua publicação, esses conteúdos nas redes sociais podem ser classificados como verdadeiros, falsos ou não verificados com base na autenticidade dos fatos apresentados (BIAN et al., 2020). Os primeiros estudos sobre detecção de conteúdos falsos focavam na construção de características de conteúdos que poderiam facilitar a detecção do fato, incluindo o texto, dados de perfil de usuário e padrões da propagação (YUAN; MA; ZHOU, 2020). Padrões da linguagem utilizada, como estilo da escrita e manchetes sensacionalistas (KWON et al., 2013), análises lexicais e sintáticas (POTTHAST et al., 2017), já foram utilizados para separar conteúdos falsos de verdadeiros. Além da linguagem, alguns estudos evidenciaram características do usuário (CASTILLO; MENDOZA; POBLETE, 2011) e temporais (KWON et al., 2013) com base nos primeiros momentos em que o conteúdo foi vinculado. No entanto, trabalhar com esse dado armazenado em uma estrutura tabular tradicional pode con-

sumir muito processamento e não ser a forma mais eficiente (PATTANAIK; MANDAL; TRIPATHY, 2023b). Além disso, essas características são facilmente manipuladas pelos usuários para parecer o mais verídico possível.

Ao longo dos últimos anos, os pesquisadores criaram vários modelos de inteligência artificial, utilizando técnicas de processamento de linguagem natural, aprendizado de máquina e algoritmos de aprendizado profundo (PATTANAIK; MANDAL; TRIPATHY, 2023b). Porém, nos últimos anos, os algoritmos de aprendizado profundo têm alcançado conquistas incríveis na detecção de rumores. Muitos pesquisadores têm extraído características visuais, de propagação e do histórico social do usuário, combinando múltiplos fatores para alimentar um sistema de detecção mais robusto (TAN et al., 2023). Sendo assim, considerando a evolução do tema, o escopo desta monografia tem foco nos algoritmos de aprendizado profundo.

A evolução nos últimos anos para resolver consistiu na aplicação de diversas redes neurais para identificar representações de alto nível baseadas nos grafos das interações dos usuários. Redes neurais recorrentes (RNN) (MA et al., 2015), redes neurais convolucionais (CNN) (YU et al., 2017) e, mais recentes, as redes neurais de grafo foram aplicadas nesse contexto da difusão do grafo de propagação do conteúdo (YUAN et al., 2019). Porém, esses métodos não têm como foco a detecção precoce do conteúdo falso, considerando que é necessário um tempo até que a informação se propague por interações e re-postagens (YUAN; MA; ZHOU, 2020).

Nesse sentido, o grande desafio da detecção precoce está na quantidade limitada de informações nos primeiros momentos após a postagem, o que leva ao questionamento: será que a propagação inicial fornece informação suficiente para detectar corretamente a veracidade do rumor? Alguns estudos como (SONG et al., 2019) e (YUAN et al., 2019) exploram a detecção precoce de notícias falsas, estabelecendo com um número mínimo de interações. Porém, segundo (YUAN; MA; ZHOU, 2020) a principal limitação desses métodos é que eles ignoram a importância do histórico dos usuários autores na detecção precoce.

Seguindo a ordem cronológica de evolução do tema, percebe-se o aumento da quantidade de informação utilizada, porém, isso traz o problema da disponibilidade de conjuntos de dados para testar esses métodos. Nos trabalhos anteriores a 2021 é comum encontrar a utilização de três principais conjuntos de dados, *Twitter 15* e *Twitter 16* (MA; GAO; WONG, 2017) e o *Weibo 15* (MA; GAO; MITRA, 2016), em que a maioria dos trabalhos que aplicaram redes neurais alcançaram uma acurácia nos dados de teste superior a 90%, especialmente os que utilizaram o grafo de interações dos conteúdos.

Com relação ao tempo necessário para atingir esse nível de acurácia na detecção, os melhores resultados ficam entre 3 e 4 horas. Assim, há um espaço na pesquisa para que esses trabalhos que tiveram um bom desempenho no passado recente possam ser testados em diferentes conjuntos de dados e idiomas, também que inovações possam ser adicionadas para reduzir o tempo de detecção do conteúdo falso que é tão crítico quanto à assertividade.

Outro fator observado na literatura é que a maioria dos estudos que apresentam um bom desempenho nos *datasets Twitter 15*, *Twitter 16* (MA et al., 2015) e *Weibo* (MA; GAO; MITRA, 2016), mesmo os que disponibilizam o código, já utilizam a forma pré-processada desse dado, com pouca informação de como foi feito o processamento a partir do dado bruto. Isso abre espaço para o desenvolvimento de estudos em outros conjuntos de dados disponíveis recentemente, em destaque o PHEME (KOCHKINA; LIAKATA; ZUBIAGA, 2018) que apresenta *tweets* relacionados a eventos reais com grande impacto.

1.1 Motivação

No cenário da área de pesquisa, observando os trabalhos anteriores, percebem-se algumas lacunas que podem ser preenchidas. Os algoritmos que apresentaram bons resultados no passado devem ser testados nos novos conjuntos de dados disponíveis para entender de fato se conseguem generalizar em conteúdos de contexto distinto. O tema da detecção precoce é abordado raramente, fator que é tão importante quanto a assertividade na detecção do rumor, pois de nada adianta atestar a veracidade de um conteúdo depois que seu efeito destrutivo já foi alcançado. No desenvolvimento de novos algoritmos e comparação de resultados, há uma lacuna para a padronização de experimentos, visto que com relação à avaliação da detecção precoce, não há um consenso com relação às métricas que devem ser apresentadas e como apresentá-las de forma temporal para comparar diferentes métodos em condições similares, para que o leitor encontre na literatura qual ferramenta se adapta melhor a cada caso, visando se aproximar do cenário real.

No cenário social há uma necessidade no combate à disseminação de desinformação. As redes sociais permitem o compartilhamento de informações e notícias em tempo real, assim o impacto de qualquer conteúdo enganoso, informações ou fotografias não confirmadas deve ser reconhecido e coibido, pois se espalham pela sociedade, causando instabilidade e ansiedade. Como exemplo, *politifact.com* é um site onde notícias, histórias e eventos recentes são classificados como rumores ou não. Como mostrado na Figura

1, em 11 de agosto de 2022 foi publicado que *Ashton Kutcher* perdeu supostamente a capacidade de ouvir, ver e andar após receber a vacina do COVID-19. Mas alguns dias depois, o ator fez o anúncio oficial de que, antes da vacina contra COVID-19, ele havia sido diagnosticado com vasculite. Depois desse comunicado, o site declarou que esse rumor era falso. Porém, esse tipo de conteúdo causou um impacto negativo nas pessoas que estavam para receber a vacina (PATTANAIK; MANDAL; TRIPATHY, 2023b).

Figura 1 – Rumor falso sinalizado pelo site *Politifact.com* (PATTANAIK; MANDAL; TRIPATHY, 2023b)

1.2 Objetivos e Desafios da Pesquisa

O objetivo deste estudo é avaliar o desempenho de redes neurais de grafo na detecção precoce de rumores em publicações textuais de redes sociais. O resultado terá o comparativo do desempenho de arquiteturas GNN com os métodos mais utilizados na literatura, avaliando se a representação em grafo utilizada na arquitetura GNN aumenta a capacidade de detecção de rumor com o passar do tempo, observando se o custo de processar o grafo da propagação é válido com relação ao tempo de detecção.

Este objetivo se desdobra em alguns objetivos específicos, como:

- ❏ Desenvolver um método padronizado para comparar algoritmos de detecção de rumores.
- ❏ Investigar o comportamento das redes neurais de grafos em comparação com algoritmos tradicionais na detecção de rumor.
- ❏ Analisar a detecção de rumor de forma temporal, do início ao fim em eventos reais propagados no *Twitter*

- ❑ Apresentar uma comparação justa entre modelos tradicionais e GNN's determinando o algoritmo mais completo com base em resultados experimentais padronizados.

1.3 Hipótese

Conforme a literatura, o uso de aprendizado profundo para detecção de rumor tem contribuído significativamente para o avanço da área, em comparação com os modelos de aprendizado de máquina tradicionais (PATTANAIK; MANDAL; TRIPATHY, 2023b). Porém, há algumas perguntas que são importantes serem respondidas, principalmente quando pensamos no cenário real. A maioria dos estudos é focada no aumento na assertividade na detecção do rumor, processando uma quantidade maior de informações e extraindo características mais complexas do dado. Porém, é importante mencionar que ter um modelo com boa assertividade que precise de muito tempo desde a publicação do rumor para a identificação vai permitir o efeito destrutivo da desinformação. Parte-se da premissa que é essencial detectar qualquer tendência ou rumor potencial o mais cedo possível e seguindo a cronologia evolutiva de técnicas de detecção de rumor em que o algoritmo de aprendizado profundo com resultados mais promissores recentes são as redes neurais de grafos, pois permitem o ganho de informação com o processamento do dado de propagação do conteúdo, ou seja, interações entre os usuários, e que há poucas conclusões em relação à redução do tempo de detecção. Sendo assim, a hipótese desse trabalho é definida como: Redes neurais de grafos permitem a detecção de rumores com desempenho melhor ou equivalente àquela obtida por algoritmos tradicionais, porém em um tempo menor desde a publicação do conteúdo. Isso pode ser testado e comprovado mediante um método padrão que compare diferentes algoritmos em condições similares em um mesmo conjunto de dados.

1.4 Contribuições

Em resumo, diferentemente da maioria dos trabalhos de detecção de rumores que avaliam conjuntos de dados estáticos, esta abordagem introduz um método dinâmico e sensível ao tempo, refletindo melhor a propagação dos rumores em um cenário real. As principais contribuições são as seguintes:

- Treinamento e teste temporal para simular cenários do mundo real. Muitos trabalhos de detecção de rumor utilizam divisões aleatórias dos dados para treinamento e teste, o que pode causar vazamento de informações, ou seja, o algoritmo pode ser treinado com um dado que propagado no futuro e posteriormente avaliado com um dado do passado. Em contraste, nesse trabalho é aplicada uma divisão cronológica rigorosa, garantindo que o modelo seja treinado apenas com dados que estariam disponíveis a partir de um intervalo de tempo definido, simulando de forma mais precisa as condições do mundo real.

- Modelagem de grafos heterogêneos: os avanços recentes na área de redes neurais de grafos são notáveis, principalmente no contexto de redes sociais. Assim, este trabalho utiliza o aprendizado baseado em grafos, estruturando interações em redes sociais como um grafo heterogêneo, estrutura que reflete de forma mais fiel as interações em redes sociais reais, permitindo que o modelo capture relações entre diferentes entidades.

- Avaliação do desempenho de detecção de rumor no tempo: outra lacuna observada na literatura é avaliar o desempenho de algoritmos de detecção de rumor em um único ponto no tempo ou após a formação completa da rede de propagação. Nossa estrutura introduz um módulo de inferência temporal, que avalia as previsões do modelo em múltiplos intervalos de tempo (por exemplo, a cada 10 minutos), observando como o algoritmo se comporta do início ao fim de uma propagação de rumor.

1.5 Organização da Dissertação ou Tese

Este trabalho foi organizado em seis capítulos: introdução, fundamentação teórica, trabalhos relacionados, proposta, experimentos e conclusão. A introdução apresenta o contexto em que o tema da pesquisa se encontra, sua importância e motivação. Também é mencionado quais hipóteses foram formuladas e objetivos que se pretendem alcançar ao final dos experimentos. A fundamentação teórica explica o que é desinformação e seu subconjunto, os rumores, que são o foco desse trabalho. São apresentados e comparados vários trabalhos de detecção de rumor, especialmente os que contêm foco em detecção precoce, e por fim, são explicadas as definições das técnicas que serão utilizadas no método proposto. O capítulo 5 mostra como foi construído o método para comparativo de algoritmos de detecção de rumor com foco em detecção precoce, chamado aqui de

ERDB, necessário para realizar os experimentos que estão no penúltimo capítulo, que também mostram os resultados da pesquisa. Por fim, a conclusão apresenta as principais contribuições e sugestões para a evolução do tema.

Capítulo 2

Fundamentação Teórica

2.1 Desinformação

Antes de apresentar as definições de rumor ou notícia falsa *(fake news)* é necessário falar sobre o conceito de desinformação: informações falsas que são deliberadamente disseminadas com o intuito de enganar as pessoas (ISLAM et al., 2020). Dados imprecisos, sem base factual e com a intenção de manipular a opinião pública também são classificados como desinformação. Outra via de desinformação pode surgir quando alguém opta por omitir fatos e, em seu lugar, espalha inverdades. Em resumo, expressões como notícias falsas, rumor e *spam* são todos variações do mesmo fenômeno (WU et al., 2019), como mostrado na Figura 2.

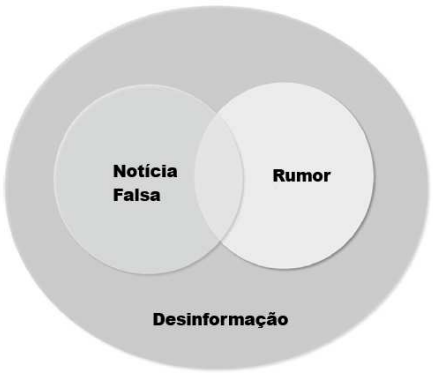

Figura 2 – Desinformação e suas variações

Neste trabalho, o foco é na detecção de rumor, porém, antes, é necessário entender

o que diferencia esses dois tipos de desinformação.

- Notícias falsas *(fake news)* são conteúdos criados de forma proposital e apresentados como se fossem reais, visando enganar o público (PATTANAIK; MANDAL; TRIPATHY, 2023b). Um exemplo disso, são as notícias falsas publicadas em 2025 em sites falsos travestidos de portais conhecidos no Brasil, os quais informavam sobre a segunda edição do Concurso Público Nacional Unificado CPNU, com objetivo de enganar candidatos e roubar dados pessoais (Brasil. Secretaria de Comunicação Social, 2025).

- Rumores são definidos como informações não confirmadas, de credibilidade duvidosa, espalhadas por alguém (PATTANAIK; MANDAL; TRIPATHY, 2023b), geralmente um usuário de rede social, podendo ter muitos seguidores ou não. O fato de ter muitos seguidores ajuda na propagação mais rápida do rumor. Exemplos desse tipo de desinformação são vários, desde ataques a candidatos a processos eleitorais à divulgação de cura de doenças por métodos não comprovados pela ciência.

A intersecção vista na Figura 2 entre rumores e notícias falsas pode acontecer quando uma notícia falsa é compartilhada por um usuário comum de rede social. O cenário oposto, em que uma notícia falsa utiliza um ou mais rumores dentro do seu conteúdo pode acontecer, mas é menos frequente.

2.2 Propagação de rumor

Rumores têm grande potencial de se espalhar rapidamente pelas redes sociais, causando impactos significativos tanto no aspecto econômico quanto no social. A Figura 3 ilustra um exemplo de propagação de um rumor no antigo *Twitter*, atual *X*.

A mensagem original iniciou uma alegação sobre as circunstâncias da motivação do assassinato do garoto *Michael Brown*, publicada pouco tempo após o ocorrido. Ela afirmava que ele havia sido baleado dez vezes pela polícia por roubar doces. Essa mensagem foi compartilhada por diversos usuários na plataforma, e em menos de 24 horas, cerca de 900 mil usuários já estavam envolvidos — seja repostando, comentando ou questionando a veracidade da publicação original.

Na Figura 3, é possível observar que o usuário 7 chegou a questionar a veracidade da mensagem inicial. Caso o boato tivesse sido identificado e refutado de forma rápida, sua propagação poderia ter sido contida de maneira mais eficaz.

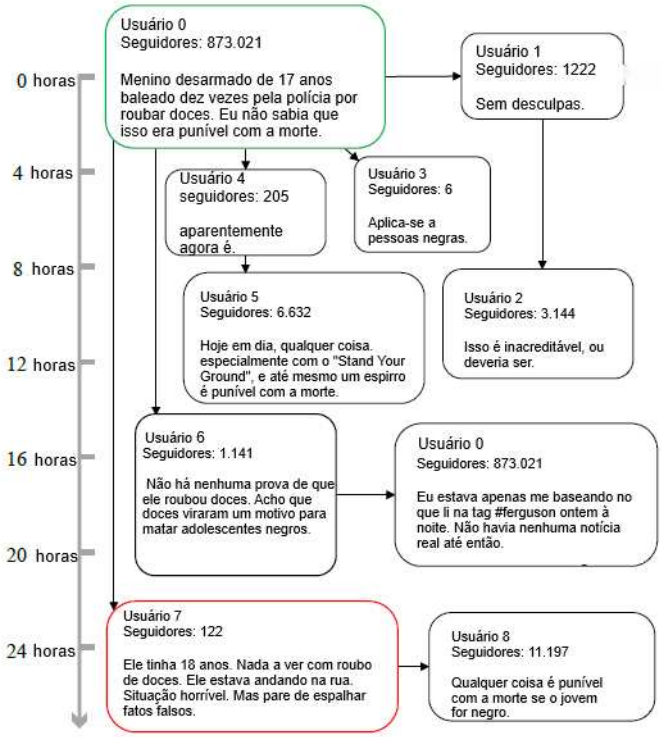

Figura 3 – Exemplo de propagação de Rumor
Adaptado de (ZHOU et al., 2019)

2.2.1 Detecção Automática de Rumores

Dada a velocidade com que rumores podem se espalhar nas redes sociais, é um requisito das ferramentas de detecção serem rápidas e automáticas. Essa tarefa é normalmente tratada como um problema de classificação binária, onde o objetivo é rotular publicações como verdadeiras ou falsas com base em seu conteúdo textual, atributos do usuário e interações.

A construção de modelos de classificação requer um conjunto de dados rotulado previamente, com confiabilidade. Esses dados são geralmente obtidos a partir de bases públicas, em que postagens foram previamente verificadas por pesquisadores ou jornalistas.

Há ainda a etapa de pré-processamento desse dado que inclui atividades como limpeza de conteúdo textual, transformação dos dados textuais em vetores numéricos e preenchimento de dados faltantes.

2.3 Processamento de linguagem natural

Uma das informações mais importantes que algoritmos de detecção de rumor devem processar são textos. E para conseguir generalizar e extrair dados relevantes, várias etapas de pré-processamento são aplicadas, além de que, no final, uma representação numérica é gerada para que os modelos consigam aprender com essas informações. As principais etapas de limpeza textual geralmente são:

- *Tokenização*: separação de um texto em unidades menores, como palavras ou frases.
- *Stopwords cleaning*: eliminação de palavras comuns sem valor semântico relevante, como “de”, “o”, “e”.
- *Stemming* e *Lemmatização*: técnicas para reduzir palavras à sua forma base ou raiz.
- *Lowercasing*, remoção de pontuação e normalização: procedimentos adicionais para uniformização dos dados.

2.3.1 Representação textual

Para que os modelos de *machine learning* possam aprender com dados textuais, é necessário transformá-los em vetores numéricos. Entre as técnicas disponíveis na literatura, nesse trabalho, optou-se por utilizar um *transformer* pré-treinado, o GloVe *(Global Vectors for Word Representation)* (PENNINGTON; SOCHER; MANNING, 2014), que combina a eficiência do *Word2Vec* com o modelo estatístico de coocorrência global das palavras. A principal motivação do GloVe é que as frequências de coocorrência entre palavras em um corpus contêm informações semânticas relevantes. Por exemplo, palavras como *rei*, *rainha* e *trono* aparecem frequentemente em contextos semelhantes, sugerindo relações semânticas próximas. GloVe busca capturar essas relações por meio de vetores densos de baixa dimensão.

A construção do GloVe inicia-se com uma matriz de coocorrência $X \in \mathbb{R}^{V \times V}$, onde V representa o tamanho do vocabulário, e X_{ij} é o número de vezes que a palavra j aparece no contexto da palavra i.

A função objetiva do modelo é definida como:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}) \right)^2 \quad (1)$$

onde:

- w_i e $\tilde{w}_j$ são os vetores de *Embeddings* da palavra e de seu contexto, respectivamente;
- b_i e $\tilde{b}_j$ são termos de viés;
- $f(x)$ é uma função de ponderação que limita o impacto de coocorrências muito frequentes ou raras

O GloVe apresenta diversas vantagens por ser pré-treinado, sendo reutilizado em diversos modelos supervisionados, mas pode requerer grande memória para textos muito grandes.

2.4 Algoritmos tradicionais para classificação de rumor e *fake news*

Diversos algoritmos de aprendizado de máquina têm sido empregados para a detecção de rumores e *fake news*. Nesta seção, apresentamos todos que serão utilizados nos experimentos, abordando 2 tipos de algoritmos do tipo "conjunto" (*bagging* e *boosting*), 1 rede neural recorrente, 2 redes neurais de grafo, sendo uma delas já construída para trabalhar com grafos heterogêneos.

2.4.1 Light GBM

O algoritmo *Light Gradient Boosting Machine (LightGBM)* (KE et al., 2017) é um modelo de árvores de decisão com *boosting* por gradiente que se caracteriza por seu método de crescimento de árvores baseado em folhas. Esse método prioriza as folhas que mais reduzem a perda, otimizando as divisões e tornando o processo de treinamento mais eficiente. Dessa forma, o *LightGBM* trabalha bem com grandes volumes de dados e alta dimensionalidade.

Para aprimorar o treinamento, o *LightGBM* inclui o algoritmo de amostragem *Gradient-based One-Side Sampling* (GOSS), que prioriza amostras com gradientes maiores, ignorando aquelas com gradientes pequenos, sob a premissa de que estas já apresentam erros

reduzidos. Segundo Ke et al. (2017), o GOSS reduz o número de instâncias de dados priorizando aquelas com maiores gradientes, que têm maior impacto no cálculo do ganho de informação, enquanto descarta instâncias com gradientes menores. Isso permite uma estimativa precisa do ganho de informação com um conjunto de dados reduzido, otimizando o processo de treinamento.

Adicionalmente, o LightGBM utiliza o método *Exclusive Feature Bundling (EFB)* para lidar com a alta *esparsidade* dos dados, ao combinar atributos mutuamente exclusivos de maneira quase sem perdas. O EFB reúne variáveis que dificilmente assumem valores distintos de zero ao mesmo tempo, um único grupo, o que reduz de forma expressiva a quantidade de atributos sem prejudicar a precisão na escolha dos pontos de divisão (KE et al., 2017). Embora a definição do agrupamento ótimo seja um problema *NP-difícil*, o *LightGBM* adota uma estratégia abrangente que, na prática, costuma oferecer uma boa aproximação. A integração de técnicas como o *GOSS* e o *EFB* faz do *LightGBM* uma solução altamente eficiente para análise de grandes volumes de dados, diminuindo tanto o custo computacional quanto o uso de memória em relação a outras implementações de *GBDT*, como o XGBoost (ALZAMZAMI; HODA; SADDIK, 2020).

Essa vantagem do *LightGBM* em comparação aos outros algoritmos baseados em árvores com relação ao tempo de treinamento se deve muito ao crescimento de árvores na forma vertical. Como todo método de árvores de decisão, possui uma raiz e folhas que podem se expandir tanto vertical quanto horizontalmente. A Figura 4 ilustra esse funcionamento, mostrando que quando a busca se encontra na folha da esquerda (círculo verde), em vez de avançar para a folha mais à direita (círculo azul), o *LightGBM* expande a partir da folha que apresenta a maior perda, crescendo de modo vertical, isto é, folha por folha. Em contraste, outros algoritmos realizam o crescimento de maneira horizontal, isto é, por níveis. Dessa forma, o LightGBM se mostra eficiente no processamento de grandes volumes de dados; contudo, em bases menores pode ocorrer *overfitting*. Sua principal vantagem está no baixo custo computacional, uma vez que consome pouquíssima memória para processar milhares de registros, mantendo ao mesmo tempo, bom desempenho nos resultados (AZRI et al., 2021).

Figura 4 – Crescimento de árvores *Light GBM* (KE et al., 2017)

De acordo com Ke et al. (2017), uma das principais características do *LightGBM* é sua estrutura de *gradient boosting*, em que o classificador base adotado é uma árvore de decisão. Para um conjunto de treinamento, como descrito na Equação 2, em que x representa o conjunto de atributos e y o rótulo, o algoritmo inicia com uma função constante $F_0 = \gamma$ e uma função de perda diferenciável.

$$X = \{(x_i, y_i) \mid x_i \in \mathbb{R}^2,\ y_i \in \mathbb{R},\ k \geq 1,\ |X| = n\} \tag{2}$$

Em que:

- X: Conjunto de pares (x_i, y_i);
- (x_i, y_i): Cada elemento é um par ordenado;
- $x_i \in \mathbb{R}^2$: x_i é um vetor no espaço bidimensional real;
- $y_i \in \mathbb{R}$: y_i é um valor real associado ao vetor x_i;
- $k \geq 1$: Condição adicional (não ligada diretamente aos pares, mas podendo ser um parâmetro do problema);
- $|X| = n$: O conjunto X possui n pares (x_i, y_i).

O *LightGBM* emprega uma função de perda que combina o erro de predição com um termo de regularização associado à complexidade das árvores, conforme mostrado na Equação 3. O algoritmo adota a estratégia baseada em *histogramas* para determinar os melhores pontos de divisão, agrupando valores contínuos em *bins* discretos, o que reduz a complexidade computacional. Além disso, implementa otimizações para lidar com dados esparsos, como a remoção eficiente de valores ausentes no cálculo do ganho de perda. Essas melhorias permitem que o *LightGBM* seja até 20 vezes mais rápido que o *GBDT* tradicional, mantendo desempenho comparável(KE et al., 2017).

$$L_{xgb} = \sum_{i=1}^{N} L(y_i, F(x_i)) + \sum_{m=1}^{M} \Omega(h_m) \tag{3}$$

em que a função de regularização é definida por:

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{4}$$

onde:

- T: número de folhas da árvore;
- w: valores (*scores*) atribuídos às folhas;
- γ: parâmetro que controla a simplicidade da árvore;
- λ: fator de regularização que controla a magnitude dos pesos.

2.4.2 *Random Forest*

Random Forest é um método de aprendizado de máquina que combina várias árvores de decisão para formar um modelo mais robusto, utilizado geralmente para tarefas de classificação e regressão. Apresentado por (BREIMAN, 2001) como uma melhoria sobre os conhecidos métodos de *bagging* através da seleção aleatória de variáveis durante a construção das árvores, o que aumenta a diversidade do modelo e reduz o *overfitting*.

Esse algoritmo gera muitas árvores de decisão durante o treinamento e produz como resultado a classe com a maior quantidade de votos (para classificação) ou a média das previsões (para regressão) das árvores individuais. O algoritmo pode ser explicado resumidamente da seguinte forma:

1. A partir do conjunto original, D, de tamanho N, são geradas B amostras *bootstrap* D_b.
2. Para cada amostra *bootstrap*, é construída uma árvore de decisão T_b por meio de divisões recursivas dos nós.
3. Em cada nó, é selecionado aleatoriamente um subconjunto de m variáveis entre o total de M variáveis ($m \ll M$) e é escolhida a melhor divisão com base em um critério de impureza (por exemplo, impureza de *Gini* ou redução de variância).
4. As previsões de todas as árvores são agregadas, onde *vt* significa voto majoritário:

❑ **Classificação:**

$$\hat{y} = \text{vt}\left(\{T_b(x)\}_{b=1}^{B}\right) \tag{5}$$

❑ **Regressão:**

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{6}$$

Para tarefas de classificação, a *impureza de Gini* é frequentemente utilizada para avaliar a qualidade das divisões:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{C} p_i^2 \tag{7}$$

onde p_i é a proporção de instâncias da classe i no nó t, e C é o número total de classes.

A robustez do *Random Forest* decorre tanto do *bagging* (agregação por *bootstrap*) quanto da *aleatoriedade na escolha das variáveis*, o que promove um bom equilíbrio entre viés e variância e maior resistência ao *overfitting*, especialmente em conjuntos de dados com alta dimensionalidade.

Na Figura 5 é representado o processo decisório de múltiplas árvores em uma tarefa de classificação em um conjunto de dados de quatro características.

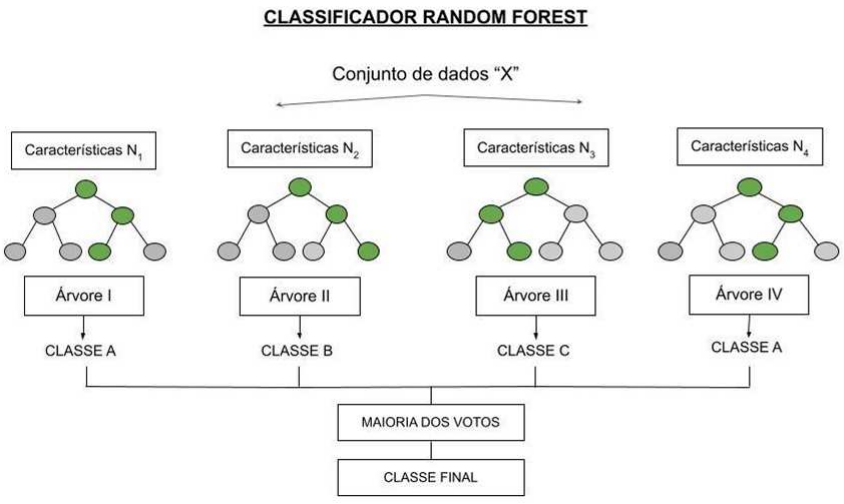

Figura 5 – Classificador Random forest

(AGUIRRE et al., 2023)

2.4.3 Long Short-Term Memory (LSTM)

As redes LSTM apresentadas por (HOCHREITER; SCHMIDHUBER, 1997) são um tipo de rede neural recorrente RNN, que são arquiteturas projetadas para lidar com dados sequenciais, como séries temporais ou texto, com a inclusão de um recurso adicional, os chamados portões (*gates*). Esse recurso pretende resolver o problema que as RNNs tradicionais apresentam em capturar dependências de longo prazo devido ao problema do *vanishing gradient*, ou gradiente desaparecendo. Esse mecanismo é composto por células de memória e três portões principais: portão de entrada, portão de esquecimento e portão de saída. Esses portões controlam o fluxo de informação ao longo da sequência, permitindo que o modelo mantenha ou descarte informações de maneira seletiva. Assim, LSTM têm a capacidade tanto de lembrar quanto de esquecer o estado anterior quando essa informação não for mais necessária, mecanismo que é executado através dos parâmetros do portão de esquecimento. Os valores do estado anterior, a memória atual e a entrada de novos dados são agregados para formar a saída da unidade LSTM (BISPO, 2018). A Figura 6 mostra todos os elementos que compõem a arquitetura

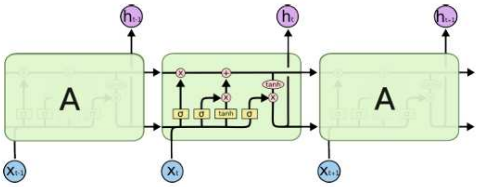

Figura 6 – Estrutura célula LSTM
(OLAH, 2015)

Uma célula LSTM é composta por três portões que controlam diferentes dados: portão de entrada (*input gate*), portão de esquecimento e portão de saída (*output gate*). Ao final de cada portão há uma função de ativação *sigmoid* σ para controlar o fluxo de informações dentro da célula.

O portão de esquecimento é mostrado na Figura 7 e recebe a entrada da célula C_t para determinar quais dos valores de índices do vetor de saída da célula anterior C_{t-1} serão mantidos. A equação dada pela função σ, que retorna valores no intervalo entre 0 e 1. W_f e b_f são os pesos e o valor *bias* para o portão de entrada (BISPO, 2018).

$$f_t = \sigma(W_f.|h_t - 1, x_t| + b_f) \tag{8}$$

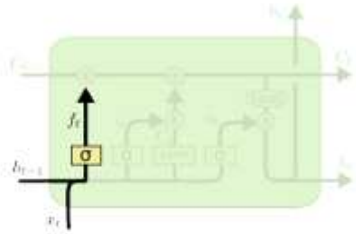

Figura 7 – *Forget Gate*
(OLAH, 2015)

A célula LSTM também calcula o que da informação de entrada será armazenado ou atualizado. Na Figura 8 essa tarefa é quebrada em duas fases. Primeiro a função σ decide quais dos dados de C_{t-1} serão atualizados no portão de esquecimento. Em sequência, um novo dado de entrada ($\tilde{C}_t$) é calculado pela função *tanh*, para então ser multiplicado com o vetor resultante do primeiro passo. Cada operação i e $\tilde{C}$ têm seus próprios pesos, que também podem ser alterados pela rede no processo de treinamento *backpropagation* (BISPO, 2018).

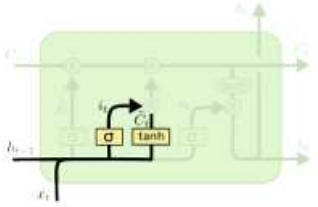

Figura 8 – LSTM *Input Gate*
(OLAH, 2015)

$$i_t = \sigma(W_i.|h_t - 1, x_t| + b_i) \tag{9}$$

$$\tilde{C}_t = tanh(W_c.|h_t - 1, x_t| + b_c) \tag{10}$$

Após essas operações, o estado da célula C_t é atualizado. As informações relevantes são atualizadas e o que não é necessário é esquecido. O estado é calculado como a seguir:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t. \tag{11}$$

Para produzir a saída final h_t da célula a função σ determina quais subconjuntos do vetor/estado C_t farão parte da saída o_t na Figura 9. Em sequência, o estado C_t é submetido a uma função hiperbólica *tanh* e multiplicado pelo resultado da operação anterior:

$$o_t = \sigma(W_o|h_t - 1, x_t| + b_0) \tag{12}$$

$$h_t = o_t \cdot \tanh(C_t). \tag{13}$$

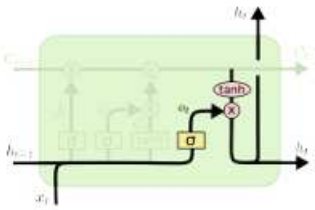

Figura 9 – LSTM *Output gate*
(OLAH, 2015)

2.5 Abordagens Baseadas em Redes Neurais de Grafo para detecção rumor

No contexto de detecção de rumor, com o avanço do campo de aprendizado profundo e da representação de dados em grafo, estruturas de interação entre usuários ou postagens que não eram possíveis de ser extraídas anteriormente com algoritmos tradicionais, agora refletem o comportamento de propagação da informação nas redes sociais. Antes de entrar no detalhe sobre GNN's é necessário apresentar alguns conceitos pré-existentes.

2.5.1 Teoria dos Grafos

Primeiro, a teoria dos grafos, que é um ramo fundamental da matemática, além de ser uma representação visual de estruturas de dados complexas, nos ajuda a compreender as relações (arestas) entre diferentes entidades (nós). Essa representação nos fornece ferramentas para modelar e analisar uma vasta gama de problemas do mundo real, como

sistemas de transporte, redes sociais e conectividade à internet (LABONNE, 2023). A notação usual para representar um grafo é $G = (V, E)$, onde G é o grafo, V é o conjunto de nós, e E é o conjunto de arestas. Os nos de um grafo podem representar quaisquer objetos, como cidades, pessoas, páginas da web ou moléculas, e as arestas representam os relacionamentos ou conexões entre esses objetos.

2.5.2 *Embeddings*

Outro conceito importante, os *embeddings*, são recursos fundamentais no aprendizado de máquina baseado em grafos. Eles transformam a estrutura complexa dos grafos em um vetor de menor dimensão e mais fácil de manipular. Essa transformação visa condensar um conjunto de dados complexo em uma forma mais simples, preservando seus padrões e relações essenciais, que servirão de base para o treinamento de redes neurais em grafos GNN's (BROADWATER; STILLMAN, 2025).

Um embedding é um vetor numérico denso que representa a identidade de um nó, aresta ou grafo, codificando informações essenciais sobre seus atributos e relacionamentos. Diversas estratégias foram propostas para gerar graph embeddings, com abordagens e aplicações distintas. Entre os métodos clássicos, destacam-se algoritmos como DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014), que utiliza caminhadas aleatórias para aprender representações baseadas na topologia da rede. Na categoria de técnicas baseadas em álgebra linear, métodos como HOPE *(High-Order Proximity preserved Embedding)* (ZHU et al., 2016) que decompõem matrizes que representam o grafo, preservando diferentes noções de proximidade estrutural. Já no contexto de *machine learning* tradicional, o *node2vec* (GROVER; LESKOVEC, 2016) introduz uma estratégia de amostragem controlada para equilibrar a exploração da estrutura global e local do grafo. Por fim, abordagens de deep learning, como as *GNN's*, modelam representações mais expressivas a partir de atributos e conectividade (KHATUA et al., 2023). As GNNs se destacam por integrarem o processo de geração das *embeddings* diretamente no próprio algoritmo de aprendizado. Em fluxos tradicionais de aprendizado de máquina, as *embeddings* costumam ser geradas como uma etapa separada, de pré-processamento de dados, funcionando como técnica de redução de dimensionalidade em tarefas como regressão ou classificação. No entanto, as GNNs fundem a criação das *embeddings* com o processo de treinamento do modelo. À medida que a rede processa os dados através de suas camadas, as representações são refinadas e atualizadas continuamente, tornando inseparáveis as fases de aprendizado e de construção das *embeddings*. Isso significa que

as GNNs aprendem, durante o treinamento, a representação mais relevante e informativa dos dados do grafo (KHATUA et al., 2023). Como exemplo, o conjunto de dados é o Political Books, que consiste em livros (representados como nós) conectados por compras frequentes em conjunto na *Amazon.com* durante o período das eleições presidenciais dos EUA em 2004 (KREBS, 2004). O uso desse conjunto de dados oferece um exemplo interessante de como o *Node2Vec* pode revelar padrões subjacentes no comportamento de compra, possivelmente refletindo agrupamentos ideológicos mais amplos entre os leitores. Esse algoritmo combina conceitos de caminhadas aleatórias *random walks* e modelos de linguagem *(Word2Vec)*. Sua principal inovação está em permitir um controle flexível da estratégia de amostragem por meio de dois hiperparâmetros: p e q. O parâmetro p regula a probabilidade de retornar ao nó anterior (*return parameter*), favorecendo uma exploração mais local, enquanto q incentiva a exploração de nós mais distantes, capturando padrões estruturais globais. Após gerar sequências de nós a partir dessas caminhadas, o *node2vec* utiliza o modelo *Skip-Gram*, originalmente proposto no *word2vec*, para aprender vetores densos que preservam relações de similaridade e conectividade entre nós. Essa abordagem torna o *node2vec* versátil, permitindo capturar tanto homofilia (nós com atributos semelhantes próximos no *embedding*) quanto similaridade estrutural (nós com papéis topológicos parecidos, mesmo que distantes na rede). Como mostrado nas Figuras 10 e 11, através dos *embeddings* é possível classificar os tipos de livros vendidos, apenas aprendendo com os dados da propagação do grafo, mostrando esse resultado por componentes UMAP de duas dimensões.

Nesse contexto, como exemplo, um *embedding* criado pelo *Node2Vec* captura a posição de um nó e seu entorno dentro do grafo utilizando informações topológicas. Isso significa que ele resume como o nó está conectado a outros, capturando de forma eficaz seu papel e importância na rede. Esses *embeddings* são poderosos porque transformam dados complexos e com alta dimensionalidade de atributos em vetores de tamanho fixo, que podem ser facilmente utilizados em diversas análises e tarefas de aprendizado de máquina. Por exemplo, sem entrar no detalhe da formulação desse vetor, para o livro, *'Losing Bin Laden'* o vetor correspondente seria:

$$\begin{bmatrix} -0.2851 & -0.2838 & 0.0777 & -0.0729 & -0.2468 & -0.3374 & 0.1027 & 0.2240 & -0.3902 & -0.0426 \\ 0.0971 & -0.1599 & -0.1009 & -0.2263 & -0.0638 & -0.2247 & 0.0773 & 0.2096 & 0.1982 & -0.2184 \\ 0.3876 & 0.1914 & 0.0255 & 0.1630 & 0.0761 & 0.3607 & -0.2749 & 0.1911 & -0.2091 & 0.2043 \\ -0.0761 & -0.0584 & 0.0909 & 0.0816 & -0.2447 & 0.1658 & 0.1028 & 0.1822 & 0.1407 & -0.0114 \\ 0.2440 & -0.0640 & -0.1073 & 0.0611 & -0.1935 & -0.0511 & 0.2298 & -0.0558 & -0.1297 & -0.1124 \\ 0.2687 & 0.1276 & 0.3322 & 0.2001 & 0.3096 & -0.1601 & -0.1627 & -0.0611 & 0.4520 & 0.2089 \\ -0.3192 & -0.1202 & -0.0077 & & & & & & & \end{bmatrix}$$

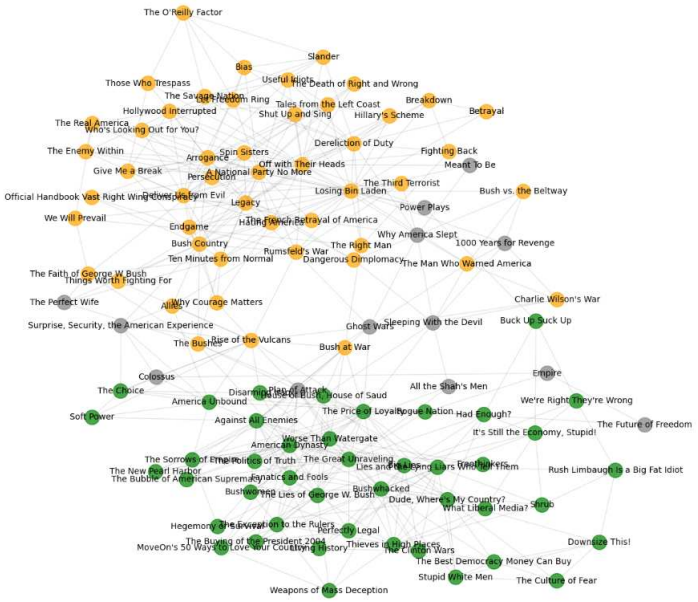

Figura 10 – Grafo - Conjunto de dados *Political Books*

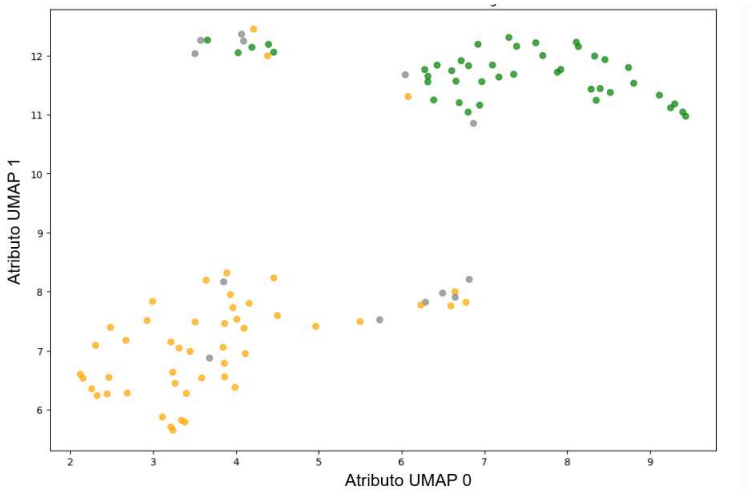

Figura 11 – *Embeddings* visualizados por meio de componentes UMAP

2.5.3 Redes Neurais de Grafo Convolucionais

O primeiro registro das GNNs tem origem no histórico das redes neurais tradicionais aplicadas a grafos. Nos anos 90, redes neurais recorrentes foram utilizadas pela primeira vez em grafos direcionados acíclicos (FRASCONI; GORI; SPERDUTI, 1998). Posteriormente, Redes Neurais Recorrentes e Redes Neurais *Feedforward* foram introduzidas nessa literatura, respectivamente por (GALLICCHIO; MICHELI, 2020), para processar grafos cíclicos.

Apesar do objetivo ser alcançado, a ideia central por trás desses métodos era construir sistemas de transição de estados em grafos e iterar até a convergência, o que limitava a capacidade de generalização e de representação dessas abordagens.

Os avanços recentes em redes neurais profundas, especialmente nas Redes Neurais Convolucionais (CNNs) (LECUN et al., 1998), resultaram na redescoberta das GNNs. As CNNs, muito utilizadas em imagens, têm a capacidade de extrair características espaciais localizadas em múltiplas escalas e combiná-las para construir representações altamente expressivas, o que levou a avanços significativos em quase todas as áreas do aprendizado de máquina e marcou o início da nova era do *deep learning* (MATHIEU; COUPRIE; LECUN, 2015).

Com base nas CNNs e *embedding* de grafos, variantes das Redes Neurais em Grafos (GNNs) foram propostas visando agregar coletivamente informações da estrutura do grafo. Dessa forma, elas conseguem modelar entradas e/ou saídas compostas por elementos e suas dependências (ZHOU et al., 2020).

A arquitetura da *Graph Convolutional Network* (GCN) é o modelo mais ideal do que se espera de uma GNN. Introduzida por (BERG; KIPF; WELLING, 2017), ela se baseia na ideia de criar uma variante eficiente das Redes Neurais Convolucionais (CNNs) aplicada a grafos. Mais precisamente, trata-se de um modelo aproximado de uma operação de convolução em grafos no contexto de processamento de sinais em grafos. Por ser versátil e de fácil uso, a GCN se tornou a GNN mais popular na literatura. De forma geral, é a arquitetura preferida para criar modelos-base sólidos em problemas que envolvem grafos (LABONNE, 2023). Como o próprio nome sugere, as GCNs podem ser compreendidas como realizando uma convolução da mesma forma que as CNNs realizam uma operação semelhante à convolução, trabalhando especialmente com imagem, como mostrado na Figura 12.

De acordo com (BERNSTEIN, 2023) a camada de convolução pode ser descrita de forma resumida e simplificada da seguinte forma:

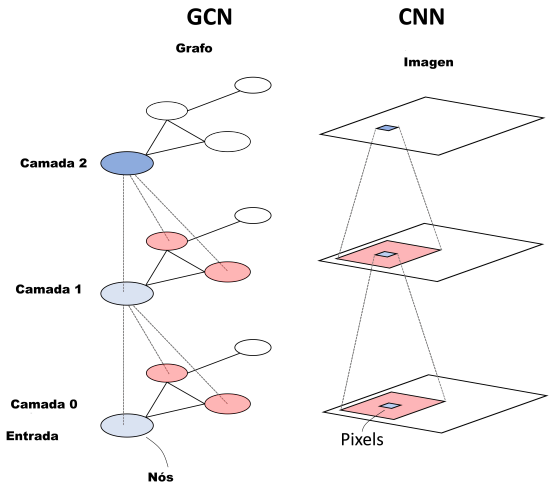

Figura 12 – GCN vs CNN

Adaptado de (BERNSTEIN, 2023)

As GCNs são compostas por camadas convolucionais empilhadas, de forma semelhante às CNNs tradicionais. Cada camada convolucional recebe como entrada os vetores dos nós provenientes da camada anterior e produz vetores de saída correspondentes para cada nó. Para isso, a camada convolucional em grafos agrega os vetores dos vizinhos de cada nó.

Por exemplo, considerando um grafo com nós (A, B, C, D), o vetor do nó A, denotado por x_A, é agregado com os vetores dos seus vizinhos, x_B e x_C. Este vetor agregado é então transformado/atualizado para formar o vetor do nó A na próxima camada, denotado por h_A. Esse mesmo procedimento é realizado para todos os nós.

Esse procedimento é frequentemente chamado de passagem de mensagens *(message passing)*, já que cada nó está enviando seu vetor aos vizinhos para que eles atualizem os seus vetores. A mensagem de cada nó é o vetor associado a ele.

Para entender a atualização e agregação da GCN poderia de forma simples.
Seja $X \in \mathbb{R}^{n \times d}$ a matriz de características correspondentes aos nós, onde:

- n é o número de nós,
- d é o número de características por nó.

Seja A a matriz de adjacência do grafo, definida como:

$$A_{i,j} := \begin{cases} 1, & \text{se existe uma aresta entre os nós } i \text{ e } j \\ 0, & \text{caso contrário} \end{cases} \tag{14}$$

As matrizes X e A são os dois dados principais de entrada de uma GCN para um grafo. Assim, a camada convolucional em grafos pode ser expressa como uma função que aceita essas duas entradas e retorna uma matriz com os vetores atualizados dos nós. Essa função é dada por:

$$f(X, A) := \sigma\left(D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}XW\right) \tag{15}$$

onde:

- $A \in \mathbb{R}^{n \times n}$: matriz de adjacência
- $I \in \mathbb{R}^{n \times n}$: matriz identidade
- $D \in \mathbb{R}^{n \times n}$: matriz de graus do grafo $A + I$
- $X \in \mathbb{R}^{n \times d}$: dados de entrada (vetores de características por nó)
- $W \in \mathbb{R}^{d \times w}$: pesos da camada
- $\sigma(\cdot)$: função de ativação (por exemplo, ReLU)

2.5.4 GAT- Redes de Atenção em Grafos

As Redes de Atenção em Grafos (GAT), *Graph Attention Networks*, considerando a tradução direta do termo em português, representam uma melhoria teórica em relação às GCNs (VASWANI et al., 2017).

Em vez de utilizarem coeficientes de normalização estáticos, a GAT propõe fatores de ponderação calculados por um processo chamado *(self-attention)*. Esse mesmo processo se tornou popular em uma das arquiteturas de aprendizado profundo mais bem-sucedidas: o transformer, em modelos como o BERT e o GPT-3 (LABONNE, 2023).

As GATs foram introduzidas por (VELIČKOVIĆ et al., 2017) e tornaram-se uma das arquiteturas de GNN mais populares.

A principal inovação por trás das GATs é que alguns nós são mais importantes do que outros, embora isso já ocorresse com a camada convolucional em grafos: nós com poucos vizinhos eram mais relevantes, graças ao coeficiente de normalização. Porém, essa abordagem é limitada porque considera apenas o grau de cada nó.

Por outro lado, o objetivo da *attention layer* é gerar fatores de ponderação que também levem em consideração a importância dos atributos dos nós.

Esses fatores são chamados de *attention scores* α_{ij}. O operador nessa camada é definido da seguinte forma:

$$h_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \mathbf{x}_j \tag{16}$$

onde:

- α_{ij} é o escore de atenção entre os nós i e j
- $\mathbf{W}$ é a matriz de pesos aprendida
- $\mathbf{x}_j$ é o vetor de entrada do nó j

Uma característica importante das GATs é que os *attention scores* são calculados implicitamente ao comparar as entradas entre si — daí o nome *(self-attention)*.

2.5.4.1 Transformação Linear

O *attention scores* representa a importância relativa entre um nó central e um nó vizinho. Esse escore requer as características dos dois nós envolvidos. Na camada de atenção em grafos, essa relação é representada por uma concatenação entre os vetores ocultos dos nós, W_{xi} e W_{xj}, $[W_{xi}||Wxj]$. Aqui W é uma matriz compartilhada para calcular os pesos das camadas ocultas. Uma transformação linear adicional é aplicada ao resultado, gerando uma matriz concatenada W_{att}, onde os pesos são aprendidos para gerar os *attention coefficients* α_{ij}

$$\mathbf{a}_{ij} = \mathbf{W}_{att}^T \left[\mathbf{W}\mathbf{x}_i \,\|\, \mathbf{W}\mathbf{x}_j \right] \tag{17}$$

A saída é então passada por uma função de ativação, como em redes neurais tradicionais.

2.5.4.2 Função de ativação

A não linearidade é um componente essencial em redes neurais. Isso é implementado na GAT aplicando a função **Leaky ReLU** à saída do passo anterior:

$$a_{ij} = \text{LeakyReLU}(\alpha_{ij}) \tag{18}$$

Porém, agora os valores resultantes não estão normalizados.

2.5.4.3 Softmax

Para comparar diferentes *attention scores* é necessário valores normalizados na mesma escala. Em *machine learning*, é comum usar a função **softmax** para esse propósito.

Seja $\mathcal{N}_i$ o conjunto de nós vizinhos do nó i, incluindo o próprio nó. Podemos definir o coeficiente de atenção normalizado α_{ij} como:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{19}$$

No entanto, há outro problema: *self-attention* pode não ser muito estável.

2.5.4.4 *Multi-head attention*

Esse problema já havia sido observado por (VASWANI et al., 2017) no artigo original do *Transformer*. A solução proposta por consiste em calcular múltiplos *embeddings* com seus próprios *attention scores*, em vez de uma única. Essa técnica é chamada de *multi-head attention*.

A implementação é direta, pois basta repetir os três passos anteriores várias vezes. Cada instância produz um *embedding* h_k, onde k é o índice. Existem duas formas de combinar esses resultados:

- **Média**: Neste caso, somamos os diferentes *embeddings* e normalizamos o resultado pelo número de *attention heads*:

$$\frac{1}{n} \sum_{k=1}^{n} h_k = \frac{1}{n} \sum_{k=1}^{n} \sum_{i \in \mathcal{N}_k} \alpha_{ik} W_k x_k \tag{20}$$

- **Concatenação**: Aqui, concatenamos os diferentes *embeddings*, o que resulta em uma matriz maior:

$$h = \Big\|_{k=1}^{n} h_k = \Big\|_{k=1}^{n} \sum_{i \in \mathcal{N}_k} \alpha_{ik} W_k x_k \tag{21}$$

Na prática, existe uma regra simples para saber qual usar: escolhemos o esquema de concatenação quando se trata de uma camada oculta, e o esquema de média quando for a saída final.

Todo o processo é resumido na Figura 13 abaixo

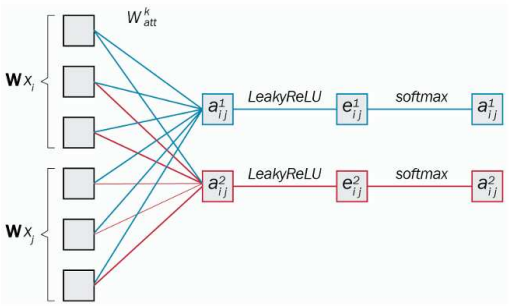

Figura 13 – Multi Head Attention
(LABONNE, 2023)

2.5.5 GNN's e grafos heterogêneos

Para o problema de detecção de rumores, a preparação dos dados para aprendizado em grafos envolve normalmente a construção de um grafo heterogêneo, o que é um pouco mais complexo, pois diferentes tipos de nós e arestas criam estruturas de grafo mais difíceis de serem aprendidas. Em particular, um dos principais desafios em redes heterogêneas é que as características dos diferentes tipos de nós ou arestas nem sempre têm o mesmo significado ou dimensionalidade — o que é exatamente o caso de redes sociais, onde temos usuários/posts criadores e usuários *tweets* que interagem com esses posts.

Um grafo heterogêneo de *tweets* e usuários, por exemplo, é denotado por $G = (V, E)$, onde V e E são o conjunto de nós e arestas, respectivamente. Os nos $V = \tau, \upsilon$ consistem no conjunto de *tweets* de origem $\tau = t_1, t_2, t_3...$ e no conjunto de usuários $\upsilon = v_1, v_2, v_3...$ que responderam aos *tweets*. As arestas $E = E_{\tau\upsilon}$ são de um único tipo: arestas tweet-usuário, que representam uma resposta.

Modelos padrão de GNNs não podem ser aplicados diretamente a grafos heterogêneos, pois as características dos nós e arestas de diferentes tipos não podem ser processadas pelas mesmas funções devido às diferenças nos tipos de atributos. Uma maneira natural de contornar esse problema é implementar as funções de mensagem e atualização separadamente para cada tipo de aresta. Durante a execução, o algoritmo precisaria iterar sobre dicionários de tipos de arestas para computar mensagens e sobre dicionários de tipos de nós para atualizar os nós.

Para evitar sobrecargas desnecessárias em tempo de execução e tornar a criação de grafos heterogêneos o mais simples possível, o *PyTorch Geometric* oferece três maneiras para que o usuário crie modelos com dados de grafos heterogêneos. O processo consiste

em pegar um modelo GNN existente e duplicar suas funções de mensagem para atuarem individualmente em cada tipo de aresta, conforme detalhado na Figura 14.

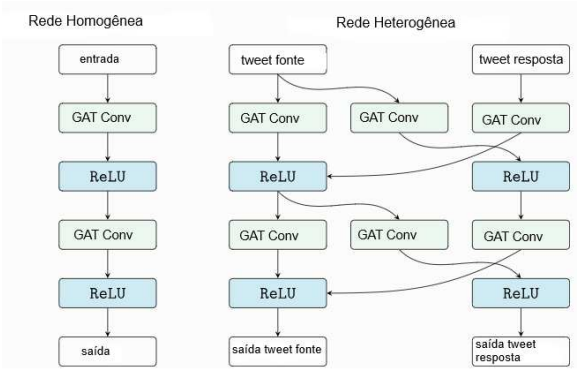

Figura 14 – Conversão de um modelo homogêneo para heterogêneo
Adaptado de (FEY, 2024)

2.5.6 Rede de Autoatenção Hierárquica (HAN)

A Rede de Autoatenção Hierárquica, *Hierarchical Self-Attention Network* (HAN) é um modelo de GNN proposto por (WANG et al., 2019). Esse algoritmo aplica mecanismos de *self-attention* em dois níveis distintos:

- **Node level attention**, que avalia a importância dos nós vizinhos em um *meta-path* específico (de forma semelhante ao funcionamento da (GAT) em ambientes homogêneos).
- **Semantic level attention**, que determina a relevância de diferentes *meta-paths*, ou seja, sequências de tipos de nós e tipos de arestas que definem um padrão relacional específico em um grafo heterogêneo. Essa é a principal vantagem do modelo, pois permite a seleção automática dos *meta-paths* mais adequados para uma tarefa específica.

Por exemplo, no trabalho de (HUANG et al., 2020), foi proposto um grafo *tweet-palavra-usuário*, conforme ilustrado na Figura 15. Os nós envolvem palavras, *tweets* e usuários. A aresta verde entre dois nós de palavras é construída com base na coocorrência das palavras. A aresta preta entre um nó de palavra e um nó de *tweet* é construída com base na frequência da palavra e na frequência da palavra no *tweet*. Já a aresta vermelha

entre um tweet de origem e um nó de usuário é construída com base no comportamento de *retweet* ou resposta a *tweets* relacionados ao tweet de origem.

A rede HAN, aplicada à tarefa de detecção de rumores, pode atribuir maior importância ao *meta-path* `tweet-usuário-tweet` do que ao `tweet-palavra-tweet` em certos cenários. Normalmente, apenas um *meta-path* está explicitamente presente no grafo, o `tweet-usuário-tweet`, mas outros tipos de nós e arestas podem ser adicionados para enriquecer a informação, já que essa GNN permite atribuir diferentes pesos a diferentes tipos de conexões entre nós.

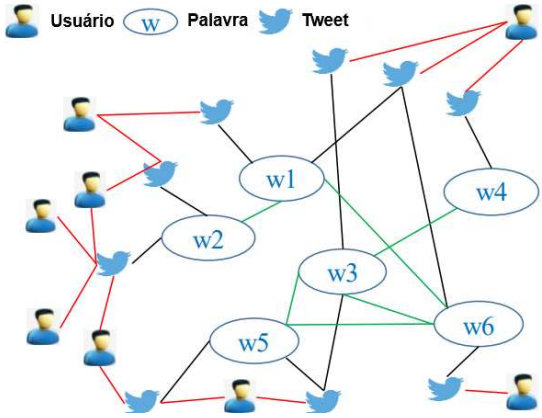

Figura 15 – Grafo heterogêneo tweet-palavra-usuário
Adaptado de (HUANG et al., 2020)

2.6 Métricas de Avaliação de classificadores binários de rumores

Neste trabalho, são utilizadas quatro métricas principais para avaliação do desempenho dos modelos no contexto de detecção de rumores. AUC, Precisão e Revocação, F1-score e AUC. Todas essas métricas estão implementadas na biblioteca *Scikit-learn* [1].

1. Precisão: Métrica que avalia a proporção de instâncias classificadas como positivas que são, de fato, positivas, ou seja, mede quantos dos exemplos classificados

[1] Scikit-learn é uma biblioteca de código aberto para aprendizado de máquina em Python, amplamente utilizada para tarefas supervisionadas e não supervisionadas (PEDREGOSA et al., 2011)

como rumor realmente são rumores. Essa métrica é essencial para acompanhar a classificação incorreta de fatos verdadeiros como rumores, o que pode levar à censura indevida.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{22}$$

Onde:

- TP: Verdadeiros Positivos (Rumores corretamente classificados)
- FP: Falsos Positivos (Fatos verdadeiros classificados como rumores)

2. Revocação *(Recall)*

   A Revocação, representa a capacidade do modelo em identificar corretamente todos os exemplos positivos, ou seja, rumores reais. Essa métrica é crucial quando o objetivo é garantir que o maior número possível de rumores seja detectado.

$$\text{Revocação} = \frac{TP}{TP + FN} \tag{23}$$

   Onde:

   - TP: Verdadeiros Positivos
   - FN: Falsos Negativos (Rumores classificados como fatos verdadeiros)

3. *Índice F1*

   O Índice F1 *F1-score* é a média harmônica entre Precisão e Revocação, sendo uma medida que busca um equilíbrio entre ambos. Essa métrica é especialmente útil em conjunto de dados desbalanceados, conforme comum em cenários de detecção de rumores.

$$\text{F1-score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}} \tag{24}$$

4. AUC - Área sob a Curva ROC

   A *AUC*, *Area Under the Curve* mede a área sob a curva ROC (*Receiver Operating Characteristic*). Ela indica o quão bem o modelo é capaz de distinguir entre as

classes positivas (rumores) e negativas (publicações verdadeiros), independentemente da probabilidade gerada pelo algoritmo escolhida. É uma métrica robusta para avaliação geral de desempenho.

$$\text{AUC} = \int_0^1 TPR(FPR^{-1}(x))\,dx \tag{25}$$

Onde:

- TPR: Taxa de verdadeiros positivos (Revocação)
- FPR: Taxa de falsos positivos ($\frac{FP}{FP+TN}$)

2.7 Considerações finais

Este capítulo apresentou os principais conceitos teóricos relacionados à desinformação, passando por algumas das técnicas tradicionais de detecção automática de rumores até as técnicas mais recentes, utilizando as Redes Neurais de Grafos (GNNs) que se destacam por sua capacidade de modelar tanto o conteúdo textual quanto a estrutura de propagação das publicações em rede.

Capítulo 3

Trabalhos Relacionados

Obter dados sobre rumores e notícias falsas tem sido uma tarefa difícil. A maioria dos pesquisadores usou os *datasets* de referência disponíveis publicamente que são *FakeNewsNet* e LIAR (WANG, 2017) para notícias falsas, TWITTER (MA et al., 2015), Weibo (MA; GAO; MITRA, 2016) e mais recentemente PHEME (ZUBIAGA; LIAKATA; PROCTER, 2016) para rumores. Coletar conjuntos de dados do mundo real não é uma tarefa difícil, porém rotulá-los adequadamente e pesquisar a veracidade de cada fato exige muito esforço. O Twitter, agora chamado "X"foi a plataforma mais usada por pesquisadores para coletar dados, talvez devido à sua facilidade de coletar os *tweets*. Porém, esse cenário mudou a partir de Junho de 2023, quando sua nova gestão bloqueou acesso livre para propósitos acadêmicos (WANG; RAJTMAJER, 2025).Alguns pesquisadores também usam sites como *Snopes, PolitiFact e Fact Check* (PATTANAIK; MANDAL; TRIPATHY, 2023b). A Tabela 1 mostra com detalhe quais os conjuntos de dados mais utilizados na literatura e suas características.

Dataset	**Rumores**	**Texto**	**Usuários**	**Tempo**	**Propagação**	**Plataforma**	**Idioma**
PHEME-R	330	y	y	y	y	Twitter	Inglês
PHEME	6,425	y	y	y	y	Twitter	Inglês
Ma-Twitter	992	y	y	y		Twitter	Inglês
Ma-Weibo	4,664	y	y	y		Weibo	Chinês
Twitter15	1,490	y	y	y	y	Twitter	Inglês
Twitter16	818	y	y	y	y	Twitter	Inglês
BuzzFeedNews	2,282	y				Facebook	Inglês
SemEval19	325	y	y	y	y	Twitter Reddit	Inglês
Kaggle Emergent	2,145	y				Twitter Facebook	Inglês
Kaggle Snopes	16,900	y				Twitter Facebook	Inglês
Facebook Hoax	15,500	y	y	y		Facebook	Inglês
Kaggle PolitiFact	2,923	y	y	y	y	Twitter	Inglês
FakeNewsNet	23,196	y	y	y	y	Twitter	Inglês
Fake.Br Brasil	7,200	y				G1 Folha SP Estadão	Português

Tabela 1 – Datasets para detecção de rumor e suas características

3.1 Revisão dos principais trabalhos e métodos detecção de rumor com *deep learning*

Todos os trabalhos desenvolvidos podem ser segmentados em 3 abordagens: redes neurais recorrentes *(RNNs)*, redes neurais convolucionais *(CNN)*, redes neurais de grafo *(GCN)*, além dos modelos híbridos que combinam mais de um método para maximizar o desempenho, que têm tido destaque na literatura recente (PATTANAIK; MANDAL; TRIPATHY, 2023a).

1. Redes neurais recorrentes: São essencialmente uma rede neural profunda do tipo *feed-forward* que processa dados sequenciais (MA; GAO; MITRA, 2016). É muito utilizada na área de processamento de linguagem natural como como tradução, reconhecimento de fala e de modelos preditivos com base em séries temporais. Para identificar rumores as suas variações são:

   - ❏ *Long short-term memory (LSTM)*
   - ❏ *Gated recurrent unit (GRU)*
   - ❏ *Bidirectional LSTM (BiLSTM)*

Um trabalho com *BiLSTM* foi proposto por (CEN; LI et al., 2022) para identificação automática de rumores. Inicialmente, os dados do texto de entrada foram usados para extrair características semânticas por múltiplas redes *BiLSTM*. Posteriormente, o modelo processa diferentes atributos, incluindo informações do usuário e do conteúdo. Para processamento do texto, dois métodos de vetorização foram utilizados, como *skip gram* e *continuous bag of words (CBOW)*. A última camada inclui uma função de ativação do tipo SoftMax para classificação. A acurácia, revocação, precisão e valor F1 do modelo foram consideradas altas nesse tipo de método, com 95,0%, 94,5%, 94,5% e 94%, respectivamente. A Figura 16 mostra de forma simplificada a estrutura do modelo.

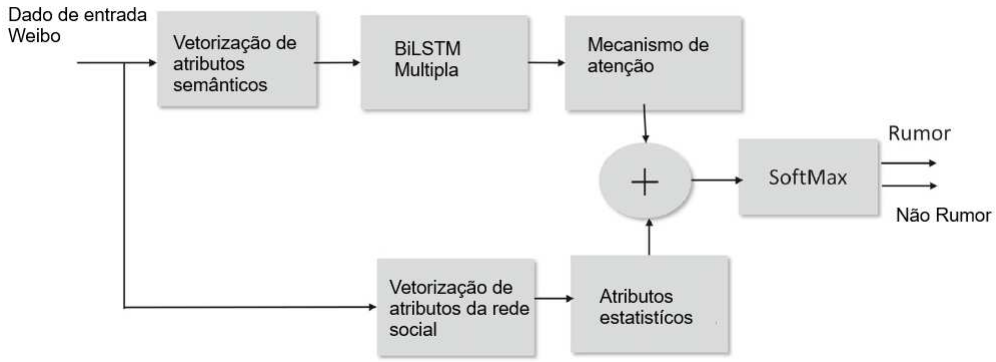

Figura 16 – Arquitetura BiLSTM
Adaptado de (CEN; LI et al., 2022)

2. Redes neurais convolucionais:

Vários pesquisadores começaram a usar recentemente *CNN* em dados de texto ou em combinação com *RNN's*. Tu et al. (2021) combinaram *CNN* com uma estrutura para incorporar atributos de texto de *tweets* e sua topologia de propagação em um único grafo. O texto original foi transformado para uma representação do tipo *word embedding* no ramo de entrada de texto. A sequência de propagação no ramo de entrada do nó foi criada identificando os usuários que compartilharam o *tweet*, então, transformando para a representação do tipo *embedding* dos nós que representam os usuários. A sequência de propagação foi alterada para que as características de mais relevância dos *tweets* de origem fossem extraídas usando CNN em ambos os ramos. O *embedding* de ambos os ramos foi então combinado para formar um único vetor. Este vetor foi propagado em mais duas camadas -

sendo uma totalmente conectada e uma camada com a função de ativação *SoftMax*, sendo o resultado obtido termos da distribuição de probabilidade sobre o de cada classe possível. Três *datasets*, incluindo *Twitter 15*, *Twitter 16* e *Weibo*, foram usados no experimento. A precisão do modelo para o Weibo foi de 95,0%, e de 79,0% e 85,0% para os *datasets Twitter 15* e *Twitter 16*, respectivamente. A Figura 17 detalha as camadas desse modelo.

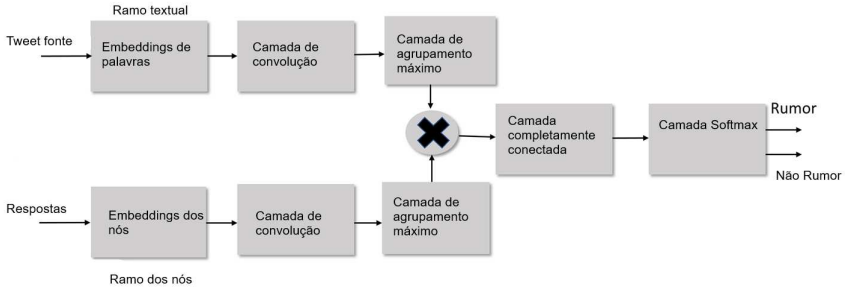

Figura 17 – Modelo Rumor2vec

Adaptado de (CEN; LI et al., 2022)

3. Redes neurais de grafo:

   As relações sociais entre diversos itens podem ser representadas em termos de dados usando a estrutura complexa de uma rede social. Um tipo de rede neural chamada rede convolucional de grafo é usada para extrair as características globais de um grafo (TAN et al., 2023). Normalmente, essas características foram extraídas da informação estrutural global obtida através da análise de todos os *tweets* em uma *thread*. Em comparação com outras redes neurais, como RNN e CNN, a GCN é mais eficaz na extração de características de nós em estruturas de redes.

   Para trabalhar com os aspectos dinâmicos, incluindo as informações heterogêneas, na detecção de rumor, (YU; ZHOU et al., 2022) apresentaram uma técnica baseada em GCN de três componentes, incluindo classificação do rumor, sua transmissão e representação. O conteúdo textual foi primeiramente representado usando o transformador *TF-IDF*, e o vetor de propagação foi então transformado usando uma matriz adjacente. Então, um grafo heterogêneo foi criado combinando os dois. A GCN, que contém uma equação diferencial ordinária (EDO), foi então analisada por este modelo. Os autores usaram o *Twitter 15* e o *Twitter 16* no experimento.

Comparado com outros modelos, apresentou uma precisão de 86,5%. Na Figura 18, que foi adaptada do grafo original, mostra um esquema resumido do modelo.

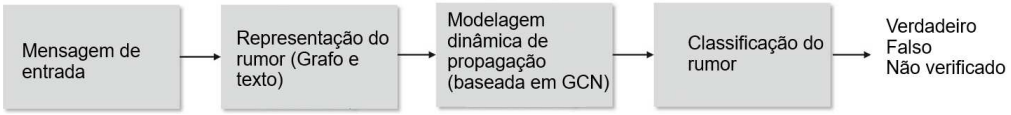

Figura 18 – Modelo GCN heterogênio

Adaptado de (YU; ZHOU et al., 2022)

É importante mencionar que muitos pesquisadores usam modelos híbridos ou conjuntos combinando duas, ou mais estratégias a fim de melhorar o desempenho do modelo durante a detecção de rumores, recurso que é bastante utilizado em modelos de *machine learning* tradicionais. O termo "híbrido" refere-se a um modelo que incorpora duas técnicas de *deep learning*, como *CNN* ou *GCN* (PATTANAIK; MANDAL; TRIPATHY, 2023b).

A tabela 2 e tabela 3 comparam a acurácia de diversos modelos baseados em deep learning, agrupados pelos conjuntos de dados mais utilizados, conforme levantamento feito por (PATTANAIK; MANDAL; TRIPATHY, 2023a). Observa-se uma melhoria significativa após a implementação de redes neurais de grafos. Em relação ao tempo necessário para alcançar esse nível de acurácia na detecção, os melhores resultados variam entre 3 e 4 horas ou não são mencionados. Isso evidencia uma oportunidade de pesquisa para que esses métodos, que apresentaram bom desempenho recentemente, sejam testados em diferentes conjuntos de dados e idiomas. Além disso, inovações podem ser introduzidas para reduzir o tempo de detecção de conteúdos falsos, o que é crucial.

Outro fator observado na literatura é que a maioria dos estudos que demonstram desempenho elevado utilizam os conjuntos de dados Twitter 15, Twitter 16 e Weibo, que já fornecem os dados pré-processados, com informações limitadas sobre como o pré-processamento foi realizado para gerar os *embeddings* e a rede de propagação. Isso abre espaço para novos estudos com conjuntos de dados brutos, especialmente o PHEME, que também inclui *tweets* relacionados a eventos do mundo real com impacto significativo na sociedade.

Tabela 2 – Comparação entre modelos de aprendizado profundo entre *datasets* comuns na literatura até 2020 (A=Acurácia, R =*Revocação*, P=Precisão e F1=F1-score)

Ano	**Arquitetura**	**Conjuntos de Dados**	**Desempenho**	**Autor**
2016	RNN	Twitter	A 91.0%	(MA; GAO, 2016)
2016	RNN	Sina Weibo	A 88.1%	(MA; GAO, 2016)
2018	RNN	Twitter	P 74.02%, R 68.75%	(CHEN et al., 2018)
2018	RNN	Weibo	P 71.7%, R 70.34%	(CHEN et al., 2018)
2019	BiLSTM, CNN	PHEME	A 86.1%	(ASGHAR et al., 2021)
2019	LSTM, CNN	Sina Weibo	A 94%, P 93%, R 95%, F1 94%	(LIU; JIN; SHEN, 2019)
2019	GCN	Twitter 15	A 77.3%	(LI; ZHANG; SI, 2019)
2019	GCN	Twitter 16	A 75.2%	(LI; ZHANG; SI, 2019)
2019	RNN	RumorEval	A 63.8%	(SUJANA; LI; KAO, 2020)
2019	RNN	PHEME	A 48.3%	(SUJANA; LI; KAO, 2020)
2020	BiLSTM	PHEME	A 91.9%	(BIAN et al., 2020)
2020	GCN	Sina Weibo	A 96.1%	(BIAN et al., 2020)
2020	GCN	Twitter 15	A 88.6%	(BIAN et al., 2020)
2020	GCN	Twitter 16	A 88.0%	(BIAN et al., 2020)
2020	CNN	YELP-2, FBN	A 87.2%,P 79.1%, R 84.7%,F1 82%	(GUO et al., 2020)
2020	Deep learning	PHEME	A 94.9%,P 37.4%, R 51.8%,F1 79%	(KOTTETI; DONG; QIAN, 2020)
2020	Deep learning, CNN	PHEME	A 94%	(WANG et al., 2019)
2020	Deep learning, LSTM	PHEME	A 81%,P 79%, R 79%,F1 81%	(WANG et al., 2022)

Tabela 3 – Comparação entre modelos de aprendizado profundo entre *datasets* comuns na literatura a partir de 2021 (A=Acurácia, R =*Revocação*, P=Precisão e F1=F1-score)

Ano	**Arquitetura**	**Conjuntos de Dados**	**Desempenho**	**Autor**
2021	GCN	Twitter 16	A 87.8%	(TU; CHEN; HOU, 2021)
2021	GCN	Twitter 15	A 85.6%	(TU; CHEN; HOU, 2021)
2021	Propagation structure, CNN	Twitter 15, Twitter 16	A 95.1%,P 94.5%, R 95.6%,F1 95.0%	(BAI et al., 2021)
2021	GCN	Twitter 15	A 89%	(KUMAR; BHATIA; SANGWAN, 2022)
2021	GCN	Twitter 16	A 91.5%	(KUMAR; BHATIA; SANGWAN, 2022)
2021	GCN	PHEME	A 69%	(KUMAR; BHATIA; SANGWAN, 2022)
2021	GCN	PHEME	A 84.1%,P 88.2%, R 95.6%,F1 89.6%	(BAI et al., 2021)
2021	Naive Bayes classifier	PHEME	A 76.7%,P 76.1%, R 76.3%	(KUMAR; BHATIA; SANGWAN, 2022)
2021	Deep neural networks	DAT@Z20, Fake News Net	A 94%, P 93%, R 95%	(AZRI et al., 2021)
2021	CNN, LSTM	ArCOV-19	A 85%,P 85%, R 85%,F1 85%	(ALSAEEDI; AL-SAREM, 2020)
2021	GCN	Twitter 15	A 83.6%	(YU et al., 2022)
2021	GCN	Twitter 16	A 86.5%	(YU et al., 2022)
2021	BiLSTM	Weibo	A 95%, P 94.3%, R 94.1%	(CEN; LI, 2022)

3.2 Detecção precoce de rumor

Embora já tenham sido realizadas pesquisas significativas focadas na assertividade da detecção de rumor, em geral, o tema da detecção precoce permanece pouco explorado. Entre os trabalhos de destaque, (LIU et al., 2022) desenvolveu um modelo de detecção de rumores online que integra representações estruturais globais da rede e ao nível dos nós. Em seus experimentos, eles avaliaram AUC no conjunto de dados *PHEME* nas primeiras 48 horas de cada evento, observando que o desempenho de todos os modelos melhorava à medida que a janela de tempo aumentava. No entanto, o estudo não especifica como o conjunto de dados foi filtrado em cada intervalo de tempo, o que limita a interpretabilidade de sua análise temporal.

No trabalho de (YANG et al., 2024), os autores propuseram um modelo chamado *Derive Domain Compensation and Association Graph Convolutional Network*, que aprimora a estrutura do domínio original do *Twitter* aproveitando dados de tópicos derivados. Eles acompanharam a acurácia do modelo nas primeiras 6 horas usando os conjuntos de dados *TWITTER* (MA et al., 2015) e *WEIBO* (WU; YANG; ZHU, 2015). Seu método demonstrou um forte desempenho de detecção já na hora zero, imediatamente após a postagem de um boato. Em (WU et al., 2024) uma estrutura de detecção precoce de rumores foi proposta com dois canais, na qual as postagens são categorizadas como “fáceis de identificar” ou “difíceis de identificar” , com base na disponibilidade de evidências online. Essas categorias são processadas em canais separados para melhorar a detecção. Para avaliar o desempenho nas fases iniciais, os experimentos incorporam linhas do tempo de propagação e marcos de data/hora como indicadores-chave. Seu modelo atinge mais de 75% de acurácia no conjunto de dados *PHEME* (KOCHKINA; LIAKATA; ZUBIAGA, 2018), com uma curva de previsão mais estável em comparação com os modelos de base, mesmo em cenários de detecção precoce.

Diante desse panorama de pesquisa, a motivação por trás do nosso trabalho é oferecer uma estrutura padronizada e reprodutível para comparar modelos de aprendizado de máquina e de aprendizado profundo, incluindo redes neurais de grafo, com foco no monitoramento de métricas-chave de detecção precoce ao longo da linha do tempo de propagação de rumor.

Capítulo 4

Método para comparativo de algoritmos

Neste capítulo, será apresentado o método criado para comparação padronizada de algoritmos de detecção de rumor, sendo explicado sob a ótica da representação de dados tabulares e de grafos. Também serão discutidas as diferenças na tarefa de detecção de rumor nesses dois tipos de dados e como o método é adaptativo em diferentes cenários.

Neste trabalho, como mostrado no Capítulo 2, foram escolhidas duas redes neurais de grafos e mais três algoritmos tradicionais para o comparativo. As vantagens e os motivos que justificaram a escolha de cada um são:

1. GNN GAT *(Graph Attention Network)*

   - ❏ Mecanismo de atenção permite que o modelo foque nos nós mais relevantes na propagação do rumor.
   - ❏ Pode integrar estrutura e conteúdo textual (via embeddings).
   - ❏ Nativa para grafos homogêneos, porém é de fácil conversão para estruturas de grafo heterogêneo.

2. GNN *HAN (Heterogeneous Attention Network)*

   - ❏ Modela grafos heterogêneos nativamente.
   - ❏ Histórico de bons resultados na literatura para capturar múltiplas formas da informação de um grafo: textual, temporal, estrutural.

3. *Random Forest - Bagging*

   - ❏ Interpretação mais fácil com geração de ranking de importância de atributos.
   - ❏ Treinamento rápido.
   - ❏ Bom desempenho em conjuntos de dados menores.

4. *LightGBM - Boosting*

   - ❏ Muito eficiente e rápido, inclusive com grandes volumes de dados.
   - ❏ Pouca customização para o treinamento, pois não requer transformação de variáveis categóricas e requer pouca normalização.

5. *LSTM* - Rede neural recorrente

   - ❏ Captura padrões temporais sequenciais, ideal para detecção precoce com base na evolução do rumor.
   - ❏ Pode ser aplicado as sequências de postagens, considerando ordem, tempo e conteúdo.
   - ❏ Pode ser mais eficaz que modelos tradicionais quando o foco é detecção progressiva e contínua.

Porém, cada método apresenta uma desvantagem que poderá ser um problema ou não, a depender do conjunto de dados e do tipo de detecção de rumor que se pretende executar. De modo geral, os principais problemas e capacidades de cada algoritmo encontrados no desenvolvimento deste trabalho estão resumidos na Tabela 4.

Tabela 4 – Desvantagens e capacidades - Algoritmos para detecção de rumor

Algoritmo	Modelagem temporal	Modelagem da propagação	Interpretabilidade	Custo computacional
Random Forest	Baixa	Não	Alta	Baixo
LightGBM	Baixa	Não	Alta	Muito Baixo
LSTM	Alta	Não	Baixa	Alto
GNN GAT	Alta	Sim	Baixa	Alto
GNN HAN	Alta	Sim	Baixa	Alto

Com relação ao custo computacional das redes neurais de grafos, esse é um problema já conhecido na literatura (BROADWATER; STILLMAN, 2025). Algumas técnicas

foram utilizadas para contornar esse problema do treinamento desses algoritmos, como o treinamento em *minibatch* em que ao invés de processar o grafo completo (*full-batch*) em cada passo do treino, é possível utilizar *minibatches* com amostragem de vizinhanças, permitindo maior escalabilidade. Também foi utilizada a técnica de *Early stopping* em que, se a métrica desejada não melhorar após um número fixo de passos consecutivos no treino *(patience)*, o treinamento é interrompido antes de rodar todos os planejados inicialmente, e os pesos da rede são ajustados para o passo anterior que apresentou a melhor métrica.

Essas são algumas abordagens fundamentais para tornar o uso de GNNs viável em cenários de grande escala, mantendo desempenho competitivo com menor custo computacional.

4.1 Método ERDB - Definição Geral

O foco desse trabalho é investigar o desempenho de detecção precoce de rumor das redes neurais de grafos comparados com algoritmos tradicionais, porém, partindo desse objetivo, foi observada a necessidade de desenvolver um método padronizado que permitisse o comparativo e análise temporal das principais métricas dessa atividade, analisando o mesmo conjunto de dados em iguais condições de treinamento e inferência. Além disso, esse método servirá de ferramenta *benchmark* para outros trabalhos, pois foi construído de forma generalista.

Para analisar a detecção de rumor sob uma perspectiva temporal e permitir uma comparação justa entre algoritmos tradicionais de aprendizado de máquina, aprendizado profundo e Redes Neurais de Grafos, foi necessário desenvolver um novo método que envolvesse no processo de detecção de rumor, do início ao fim, ou seja, desde o dado bruto até a predição final, automatizando todo o processo e gerando resultados comparativos. As principais etapas são descritas e ilustradas na Figura 19. Para dar um nome ao método, durante o texto será referido como **ERDB**, sigla para *Early Rumor Detection Benchmark*.

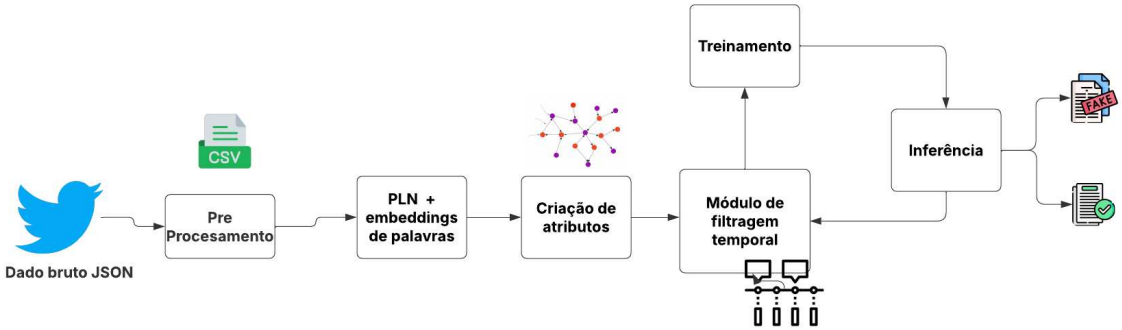

Figura 19 – Método proposto para comparar algoritmos de detecção de rumores.

4.2 Treinamento e inferência temporal

1. **Descrição do conjunto de dados:** O método foi pensado para processar dados de redes sociais incluindo os textos da publicação fonte e interações, além dos demais atributos inclusos, o que incorpora as características de boa parte dos *datasets* usados literatura para detecção de rumores em redes sociais. Estão fora de escopo o processamento de imagens, áudios e vídeos. Sendo assim, esse método se aplica a publicações de portais de notícias, *tweets* relacionados a eventos de grande repercussão, compostos por uma publicação inicial e suas respectivas interações em forma de respostas, retuítes e menções. Nessa estrutura, cada *thread* corresponde a um possível rumor e é representada como uma estrutura em árvore, na qual as interações estão associadas ao seu *tweet* original e à sequência temporal de respostas. Além das informações textuais, o conjunto pode incluir metadados como o identificador do usuário, data e hora da publicação, e atributos estruturais da conversação. Esses dados refletem a dinâmica de propagação de informações em tempo real, sendo particularmente adequados para tarefas que exigem análise cronológica e contextual, como a detecção de desinformação ou rumores em estágios iniciais de disseminação.

2. **Estruturação do dado - Mapeamento de interações e postagens:** Considerando que a maioria dos conjuntos de dados é coletado em formato hierárquico, onde cada interação faz referência à publicação e usuário fonte, nesta etapa foi desenvolvida uma *pipeline* de extração para obter os atributos de cada publicação

fonte. O resultado é um arquivo CSV onde cada linha representa uma interação reposta, com uma coluna indicando o respectivo ID da publicação e usuário fonte.

3. **NLP e pré-processamento textual:** Se houver informação textual disponível, técnicas de pré-processamento de texto são aplicadas para limpar os dados antes da geração dos *embeddings* de palavras, que são representações numéricas de linguagem natural, pois os algoritmos não são capazes de processar a informação textual na sua forma pura. Primeiramente, caracteres especiais, menções de usuários, *url* e palavras comuns sem contexto, são removidos para reduzir ruídos que podem afetar a qualidade das representações textuais. Após essa etapa de pré-processamento, os *embeddings* de palavras são gerados utilizando o modelo pré-treinado Glove (PENNINGTON; SOCHER; MANNING, 2014), garantindo que a representação numérica do texto capture relações semânticas e sintáticas. Esses *embeddings* são então utilizados como entrada para o modelo de detecção de rumores, permitindo que ele reconheça padrões linguísticos relevantes para a classificação.

4. **Criação de atributos e separação de dados:** Primeiramente, os dados são divididos em conjuntos de treinamento, validação e teste. Diferente das tarefas tradicionais de classificação, é utilizada uma divisão cronológica em vez de uma divisão aleatória. Isso é crucial porque uma divisão aleatória poderia levar a um cenário onde publicações futuras seriam usadas para prever publicações passadas dentro do mesmo evento, o que não é o comportamento desejado. Na divisão cronológica, a data e hora da publicação de origem é usada como referência. Por exemplo, se 70% do conjunto de dados for alocado para treinamento, isso significa selecionar as primeiras 70% publicações ordenados de forma crescente por tempo de publicação, garantindo que apenas postagens nas primeiras "x" horas após a postagem inicial sejam incluídas. Essa abordagem reflete melhor as condições do mundo real, onde os modelos devem fazer previsões apenas com base em dados passados. Após isso, outras tarefas precisam ser realizadas, como normalização de colunas numéricas, codificação de variáveis categóricas e imputação ou remoção de dados ausentes.

5. **Treinamento**: Nesta etapa, com todo o conjunto de dados pré-processado, o treinamento pode iniciar para ambos os dados tabulares ou grafos. Se forem grafos, os dados tabulares precisam ser convertidos para o formato específico da

biblioteca *Torch Geometric* [1]. Caso outro tipo de rede neural seja utilizada, pode ser necessário converter os dados para o formato de tensor. A partir dessa escolha do algoritmo, o treinamento não difere do processo de um classificador tradicional, exceto por uma mudança essencial:

Mesmo que as publicações estejam separadas em conjuntos de treino e teste de forma cronológica, nada impede que, por exemplo, a primeira postagem listada no conjunto de treino tenha novas respostas conforme o tempo passe, ou até mesmo tenha a última respostas ao fim de um evento.

Sendo assim é necessário o retreinamento do algoritmo a cada iteração do módulo, a partir de intervalos de tempos pré-definidos para padronização de um experimento. Isso fica mais evidente e importante quando o dado está representado na forma de grafo, por exemplo, quando uma nova publicação surge, novas arestas e novos nós só provocarão uma atualização no modelo se este for retreinado. Nos modelos que trabalham com representação tabular, normalmente isso afetará uma ou poucas colunas dependentes do tempo, como, por exemplo, número de comentários e tempo do primeiro comentário. É importante mencionar que há atributos que são temporais, mas não dependem de novas interações, como, por exemplo, se um usuário não era verificado e passou a ser.

No cenário real, esse retreinamento não precisa ser feito a cada em intervalos de tempos muito curtos. Na prática, o que pode ser feito é um constante monitoramento das métricas do modelo, com uma definição de regras de retreinamento ou utilizar a versão em tempo real desses algoritmos, *streaming*, que não está no escopo desse trabalho, mas há disponível para alguns algoritmos como, por exemplo, na biblioteca *River ML*[2] para o *Random Forest*.

6. **Módulo de filtro temporal:** Diante o comportamento desejado no treinamento, foi criada uma classe em *Python* para avaliar o desempenho dos algoritmos à medida que o tempo avança e o número de postagens/interações aumenta. Para isso, foi desenvolvido uma classe em *Python* que filtra as postagens do conjunto de teste a cada X minutos (intervalo de tempo desejado), desde que haja uma coluna de data e hora. Com o conjunto de treinamento fixo, o modelo é treinado antes da

[1] *PyTorch Geometric* (torch-geometric) que é uma biblioteca baseada em *PyTorch* para aprendizado profundo com dados estruturados em grafos. É amplamente utilizada para construção e treinamento de Redes Neurais em Grafos (GNNs) (FEY; LENSSEN, 2019).

[2] Biblioteca para modelos de *machine learning* em tempo real (MONTIEL et al., 2021)

primeira inferência, sendo que a cada x minutos até a última interação do conjunto de teste, novas inferências são feitas, avaliando o desempenho do modelo a medida que surgem novas publicações/interações. O modelo também é atualizado a cada x minutos, pois as interações das postagens do conjunto de treino podem ter novas interações conforme o tempo passa. Essa abordagem permite uma análise do desempenho do modelo no início, meio e fim da propagação em um evento com disseminação de rumor.

7. **Inferência do modelo:** A etapa de inferência registra métricas de classificação importantes como precisão, AUC, revocação e índice f1, sendo registradas utilizando a biblioteca *MLflow*. Dessa forma, as métricas de cada experimento são organizadas, registradas e posteriormente podem ser facilmente comparadas e consultadas no banco de dados do *MLflow* ou pela sua interface.

Em resumo, o módulo de treinamento, filtragem temporal e inferência são interligados, formando um *loop* que se inicia no intervalo de tempo que marca o início do conjunto de teste, passa para o treinamento e inferência, que se repete até o final do evento, permitindo que o algoritmo desejado seja testado incrementalmente.

Para deixar a explicação mais clara por meio de um exemplo prático, construído para se aproximar ao máximo do cenário real, a Figura 20 e Tabelas 5, 6 e 7 foram construídas.

Dois tipos de dados podem ser processados nesse método, a representação tradicional tabular e no formato de grafos. A Figura 20 mostra como seria a evolução do grafo de um conjunto de dados para detecção de rumor durante três intervalos de tempo distintos. Há dois tipos de nós representados pela cor verde (usuários) e pela cor azul *(tweets)*, e o que acontece durante cada intervalo de tempo é descrito como:

1. Intervalo 0: Há apenas 3 usuários e três *tweets*. Há um *tweet* sem interação com nenhum usuário (155), dois usuários interagindo com um único *tweet* e outro *tweet* que teve interação com outro usuário distinto.

2. Intervalo 1: Aqui passam a surgir novas interações, *tweets* e *usuários*. Toda nova interação ou nó no grafo foi destacado pela cor laranja. Por exemplo, o *tweet* 155 que estava sem interação, agora tem o usuário *u7* conectado a ele. Um novo usuário interage com o *tweet* 164 e aparece o novo *tweet* 160 já com a interação com o usuário *u8*.

3. Intervalo 2: Aqui já possível observar um grafo mais evoluído, com usuários interagindo com mais de um *tweet*, por exemplo, o usuário *u11*. Pode haver *tweets* isolados, pois novos conteúdos estão surgindo a todo o momento, e essa é uma característica desse tipo de grafo, gerando uma estrutura esparsa.

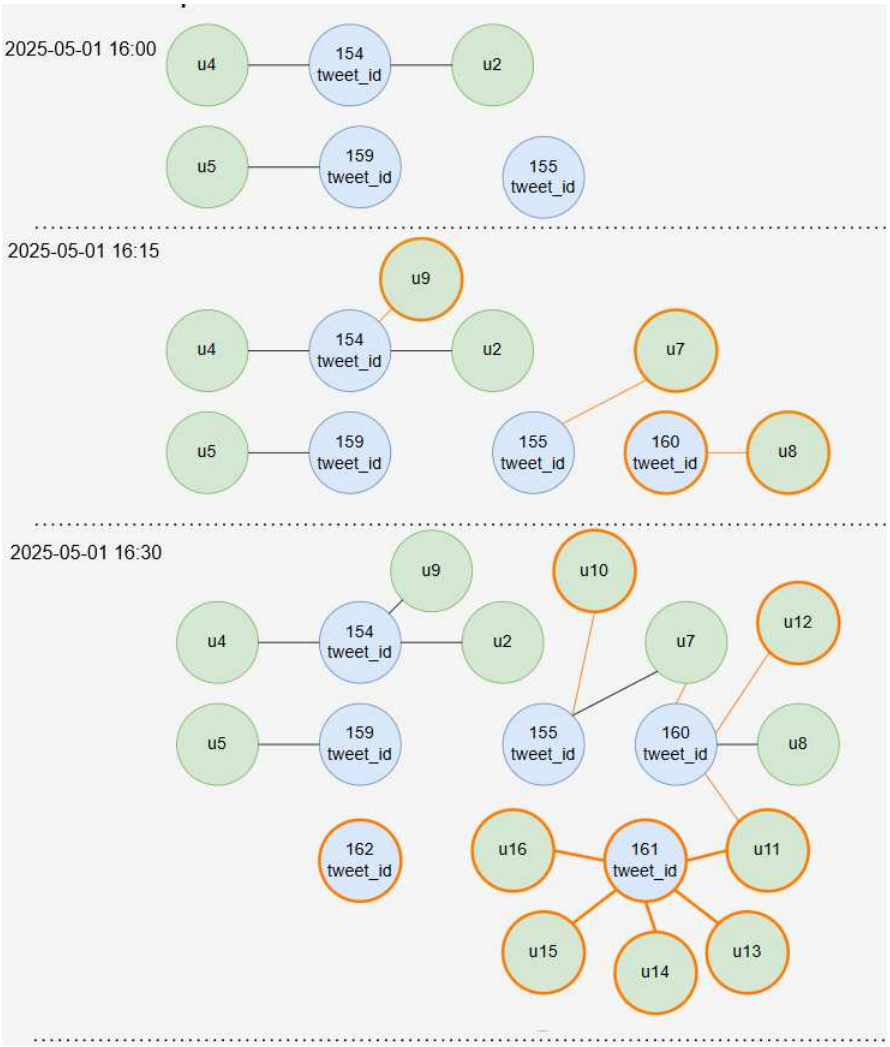

Figura 20 – Evolução temporal - Conjunto de dados em grafo

Da mesma forma, as Tabelas 5, 6 e 7 mostram essa mesma evolução temporal, considerando o dado tabular e algumas colunas de exemplo. Na Figura, os nós e arestas novos estão destacados em laranja, representando novas publicações e comentários que surgiram no novo intervalo de tempo. Nas tabelas, as novas publicações estão destacadas

com a linha toda preenchida em laranja. Na Tabela 6, há a atualização de dois atributos, destacados em laranja na sua respectiva célula. Diante deste comportamento dinâmico similar ao cenário real, fica evidenciado que a avaliação do desempenho de um algoritmo em um conjunto de dados estático é ineficaz ou pelo menos insuficiente para predizer sobre seu desempenho em um ambiente produtivo, em que atributos como número de comentários e interações entre diferentes usuários estão mudando a todo momento.

Tabela 5 – Conjunto de dados tabular: Intervalo de tempo: 2025-05-01 16:00

id	**retweets**	**likes**	**Data Hora**	***Embeddings***	**Rótulo**	**Respostas**	**Data Hora 1°resposta**
154	33	44	2025-05-01 15:43	glove	rumor	2	2025-05-01 15:47
155	21	12	2025-05-01 15:53	glove	não rumor	0	-
159	1	1	2025-05-01 16:00	glove	não rumor	1	2025-05-01 16:09

Tabela 6 – Conjunto de dados tabular: Intervalo de tempo: 2025-05-01 16:15

id	**retweets**	**likes**	**Data Hora**	***Embeddings***	**Rótulo**	**Respostas**	**Data Hora 1°resposta**
154	33	44	2025-05-01 15:43	*glove*	rumor	2	2025-05-01 15:47
155	21	12	2025-05-01 15:53	*glove*	não rumor	2	2025-05-01 16:13
159	1	1	2025-05-01 16:00	*glove*	não rumor	1	2025-05-01 16:09
160	1	1	2025-05-01 16:11	*glove*	não rumor	1	2025-05-01 16:12

Tabela 7 – Conjunto de dados tabular: Intervalo de tempo: 2025-05-01 16:30

id	**retweets**	**likes**	**Data Hora**	***Embeddings***	**Rótulo**	**Respostas**	**Data Hora 1°resposta**
154	33	44	2025-05-01 15:43	*glove*	rumor	2	2025-05-01 15:47
155	21	12	2025-05-01 15:53	*glove*	não rumor	2	2025-05-01 16:13
159	1	1	2025-05-01 16:00	*glove*	não rumor	1	2025-05-01 16:09
160	1	1	2025-05-01 16:11	*glove*	não rumor	1	2025-05-01 16:12
161	0	0	2025-05-01 16:22	*glove*	Rumor	1	2025-05-01 16:23
162	1	6	2025-05-01 16:25	*glove*	não rumor	1	-

4.3 Formulação do problema de detecção de rumor - Dados em grafos

Para o problema de detecção de rumor, a preparação dos dados para o aprendizado em grafos envolve geralmente a construção de um grafo heterogêneo, o que é mais complexo devido à presença de diferentes tipos de nós e arestas, porém permite estruturas com um ganho maior de informação.

O grafo heterogêneo publicação-usuário, utilizado neste trabalho, é denotado por $G = (V, E)$, onde V e E representam os conjuntos de nós e arestas, respectivamente. O conjunto de nós $V = \tau, \upsilon$ consiste nas publicações de origem $\tau = t_1, t_2, t_3, \ldots$ e nos usuários $\upsilon = v_1, v_2, v_3, \ldots$ que responderam às publicações. O conjunto de arestas $E = E_{\tau\upsilon}$ contém um único tipo de aresta: arestas publicação-usuário que representam interações de resposta.

A tarefa visa aprender uma função $p(c \mid t_i, G; \theta)$ que estima a probabilidade de uma publicação de origem t_i pertencer à classe c, onde c é o rótulo de categoria e θ representa os parâmetros do modelo a serem aprendidos.

Para essa tarefa, foram selecionadas duas arquiteturas de Redes Neurais em Grafos (GNN). A primeira é a *Graph Attention Network* (GAT), introduzida por (VELIČKOVIĆ et al., 2017), que melhora as Redes Convolucionais em Grafos (GCNs) ao ponderar adaptativamente os nós vizinhos, levando a um bom desempenho em diversas tarefas com grafos. Embora a GAT seja inerentemente um modelo homogêneo, ela pode ser adaptada a grafos heterogêneos utilizando transformadores apropriados que replicam as funções de mensagem para cada tipo de aresta e nó individualmente.

A segunda é a *Hierarchical Self-Attention Network* (HAN), proposta por (WANG et al., 2019), que aplica atenção automática *(self-attention)* tanto no nível de nós quanto no nível semântico. O HAN determina a relevância de diferentes meta-caminhos — ou seja, sequências de tipos de nós e arestas que definem padrões relacionais específicos em um grafo heterogêneo — capturando assim, de forma eficaz, relações heterogêneas complexas.

4.3.1 Criação do grafo heterogêneo

Grafos heterogêneos contêm diferentes tipos de informações associadas aos nós e arestas, como os diferentes tipos de nós no conjunto de dados de uma rede social, com publicação de origem e suas interações. Portanto, o conjunto de dados em formato CSV

precisa ser transformado em uma estrutura de grafo heterogêneo, utilizando a biblioteca *PyTorch Geometric*.

1. **Definição:**
   - ❏ Existem dois tipos de nós: publicação de origem e resposta.
   - ❏ As características incluem atributos da postagem de origem, (por exemplo, número de seguidores, número de curtidas e *embeddings* pré-calculados do GloVe.
   - ❏ As arestas representam cada ação de resposta a publicação de origem.
   - ❏ Os rótulos indicam se a publicação é um rumor (1) ou não-rumor (0).
   - ❏ Máscaras de entrada (*masks*) definem as divisões entre treino, validação e teste para o treinamento do modelo.
2. **Codificação *One-Hot*:** Atributos binários, como a verificação do usuário (tanto de origem quanto da resposta), são convertidos em vetores one-hot (codificação binária).
3. **Normalização (*Scaling*):** Um escalonador robusto (*Robust Scaler*) é aplicado para normalizar os atributos numéricos, reduzindo o impacto de *outliers* em variáveis como número de seguidores e curtidas.
4. **Construção do grafo:** Cada publicação e resposta recebe um índice único. O conjunto de dados em CSV contém relações entre publicações e também entre usuários. O índice das arestas (*edge index*) é criado com base nessas conexões.
5. **Divisão dos dados (*Data splitting*):** O conjunto de dados é dividido cronologicamente, conforme explicado na etapa de configuração, em treino, validação e teste. Máscaras booleanas são criadas para indicar a qual conjunto cada nó pertence.

4.4 Formulação do problema de detecção de rumor - Dados tabulares

Na detecção de rumor utilizando a representação tradicional tabular, que permite a aplicação de algoritmos de aprendizado supervisionado tradicionais, o processo é mais

simples. A partir da extração e engenharia de atributos relevantes a partir dos dados brutos, como interações entre usuários, conteúdo textual das publicações e metadados associados, estruturando-os em vetores de características.

O conjunto de dados tabular utilizado neste trabalho é representado por uma matriz $X \in \mathbb{R}^{n \times d}$, onde n representa o número de amostras (por exemplo, publicações) e d o número de atributos extraídos para cada amostra. Cada linha $x_i \in \mathbb{R}^d$ corresponde a uma publicação de origem t_i e suas características associadas — como métricas de engajamento, tempo de disseminação, padrões linguísticos, estatísticas de usuário, entre outros. O vetor de rótulos $y \in 0, 1$ indica se a publicação corresponde ou não a um rumor.

A tarefa consiste em aprender uma função de decisão $p(c \mid x_i; \theta)$ que estime a probabilidade de uma instância x_i pertencer à classe $c \in 0, 1$, onde θ representa os parâmetros do modelo supervisionado, aprendidos com base em um conjunto de dados rotulado.

A construção das *características* incluiu o uso de métricas agregadas de comportamento dos usuários, estatísticas temporais, extração de atributos linguísticos via *embeddings* e interações históricas, compondo assim uma representação expressiva para os modelos tabulares. Embora essa abordagem não capture diretamente as relações topológicas complexas dos dados como nos grafos, ela apresenta vantagens em termos de simplicidade, velocidade de treinamento e interpretabilidade dos modelos gerados.

4.5 Regularização e controle de desbalanceamento de classes

As GNNs, assim como outros modelos de aprendizado profundo, estão sujeitas ao problema de *overfitting*, (sobreajuste) especialmente quando treinadas para tarefas de classificação de nós com poucos exemplos da classe positiva, que pode ser o cenário real de detecção de rumor. Para mitigar esse problema, algumas técnicas de regularização foram implementadas, sendo elas:

1. *Dropout*: Técnica clássica de regularização utilizada em redes neurais profundas, que também se mostra eficaz em GNNs. Ele consiste em desativar, de forma aleatória, uma fração pré-definida dos neurônios durante o treinamento, impedindo que o modelo dependa excessivamente de determinados caminhos de ativação. Nesse trabalho, foi utilizado o tipo *DropEdge* que remover aleatoriamente uma

fração das arestas, ou seja, comentários das postagens, do grafo completo em cada interação de treinamento.

2. *Early Stopping*: Tecnicamente, este não é um método regularização explícita. O *Early Stopping*, "parada precoce", consiste em interromper o treinamento assim que o desempenho no conjunto de validação parar de melhorar para um número pré-definido de interações. Isso evita que a rede neural se ajuste ao ruídos do conjunto de treinamento.

3. *Weight Decay*: *Weight Decay*, "decaimento de pesos", é uma técnica de regularização explícita que adiciona uma penalização ao valor dos pesos da rede durante o treinamento, evitando que os pesos cresçam demais, ajudando na estabilidade da propagação entre vizinhos.

No controle de desbalanceamento, para evitar viés nos modelos do tipo "conjunto", os pesos internos foram ajustados para dar mais importância à classe minoritária quando a taxa de desbalanceamento é inferior a 30%. Para redes LSTM e GNN, os pesos das classes foram ajustados proporcionalmente na função de perda, a fim de garantir uma comparação justa entre os modelos.

Modelos supervisionados produzem, por padrão, probabilidades entre 0 e 1 para a classe positiva. Em vez de utilizar o limiar padrão de 0,5 para converter essas probabilidades em rótulos de classe, seleciona-se o limiar que maximiza o F1-score, equilibrando precisão e revocação. Essa abordagem incentiva os modelos a serem mais conservadores, especialmente considerando que os ajustes de peso podem levar a mais falsos positivos.

4.6 Monitoramento e comparativo de métricas

No módulo de inferência, a cada interação no conjunto de teste são calculadas as métricas mais importantes no contexto de detecção de rumor, as quais são citadas abaixo, bem como o objetivo principal de monitorar cada uma.

- ❏ Precisão: Importante para medir se o modelo erra em classificar muitos fatos verdadeiros como rumor (importante para evitar censura indevida)
- ❏ Revocação: Para garantir que o maior número de rumores reais sejam detectados
- ❏ *F1-score*: Equilíbrio entre precisão e revocação, ideal em bases desbalanceadas ou quando se pretende atingir um meio-termo entre precisão e revocação

- *AUC*: Mede o desempenho geral, independentemente do limiar escolhido (probabilidade), útil para análise comparativa, o quão bem o modelo distingue as classes positivas e negativas.

Esse processo é feito integrando o código desenvolvido com a biblioteca *MLflow* [3], ferramenta aberta utilizada para rastreamento, organização e reprodutibilidade de experimentos de *machine learning*, especialmente útil em cenários onde se deseja acompanhar e comparar diversas métricas, modelos e entender quais mudanças realmente trazem melhorias.

4.7 Considerações finais

Neste capítulo, foi detalhado o método ERDB, proposto como uma abordagem padronizada e adaptável para a comparação de algoritmos de detecção de rumor. A definição geral do método e suas etapas principais — desde o treinamento temporal até a avaliação por métricas — foram exploradas com foco na aplicabilidade em diferentes estruturas de dados, sejam elas tabulares ou em grafos.

A formulação da tarefa de detecção de rumor diferente para cada tipo de dado permitiu evidenciar as particularidades e os desafios inerentes a cada representação, reforçando a necessidade de um método que fosse flexível ao nível de tipos de algoritmos para teste. Além disso, a adoção de estratégias de regularização e controle de desbalanceamento visou mitigar os efeitos de distribuições desiguais de classes, comuns em problemas reais.

Por fim, o monitoramento temporal de métricas e a padronização dos procedimentos de avaliação garantem maior confiabilidade e reprodutibilidade dos experimentos, contribuindo para um comparativo justo entre diferentes modelos. O método ERDB, portanto, se configura não só como uma ferramenta criada especificamente para este trabalho, mas também como uma contribuição relevante para o cenário de pesquisa em detecção de rumores, permitindo sua aplicação em cenários variados e com diferentes características de dados.

[3] MLflow é uma biblioteca de código aberto voltada para o gerenciamento do ciclo de vida de modelos de aprendizado de máquina, incluindo experimentação, reprodutibilidade e implantação. (ZAHARIA et al., 2018)

Capítulo 5

Experimentos e Análise dos Resultados

Neste capítulo serão apresentados os resultados dos dois experimentos conduzidos ao longo deste trabalho, bem como a caracterização do conjunto de dados escolhido. Cada experimento aborda os cinco algoritmos listados no Capítulo 4, para contemplar a representação tabular e de grafo para a detecção de rumor.

5.1 Descrição do Conjunto de Dados

Os experimentos para testar o método proposto no Capítulo 4 baseiam-se no conjunto de dados *PHEME*, detalhado nas Tabelas 9 e 8 , uma coleção amplamente utilizada de *tweets* classificados como rumores e não rumores, frequentemente referenciada em pesquisas anteriores (KOCHKINA; LIAKATA; ZUBIAGA, 2018).

Tabela 8 – Descrição dos eventos

Evento	Número de *Tweets* (%)		Número de Usuários (%)	
	Rumor	Não Rumor	Rumor	Não Rumor
Charlie Hebdo	458 (22%)	1621 (78%)	13879 (74.2%)	4821 (25.8%)
Germanwings Crash	238 (50.7%)	231 (49.3%)	1464 (50.3%)	1442 (49.7%)
Ottawa Shooting	470 (52.8%)	420 (47.2%)	3978 (51.1%)	3794 (48.9%)
Sydney Siege	522 (42.8%)	699 (57.2%)	7545 (61.8%)	4658 (38.2%)
Ferguson	284 (24.8%)	859 (75.2%)	3792 (35%)	7001 (65%)

Tabela 9 – Descrição dos dados

Campo	**Descrição**
id	Identificador único do *tweet* de origem/resposta.
reply_id	O ID do *tweet* ao qual a resposta está reagindo.
text	Conteúdo do *tweet* de origem/resposta.
user_id	ID do usuário que publicou o *tweet* de origem ou resposta.
followers_count	Número de seguidores do usuário da origem/resposta.
retweet_count	Número de *retweets* do *tweet* de origem/resposta.
favorite_count	Número de curtidas do *tweet* de origem/resposta.
verified	Indica se o usuário da origem ou resposta é verificado.
time	Momento em que o *tweet* de origem/resposta foi publicado.
rumour	Rótulo-alvo (rumor ou não-rumor).

O conjunto de dados abrange cinco eventos principais: o atentado ao *Charlie Hebdo*, o acidente da *Germanwings*, o tiroteio em *Ottawa*, o cerco em *Sydney* e os protestos em *Ferguson*. Ele inclui *tweets* publicados como notícias de última hora durante os anos de 2014–2015, especificamente relacionados a esses incidentes. O conjunto está organizado em cinco arquivos distintos, cada um correspondente a um evento, com os dados armazenados em formato JSON. Esses arquivos contêm detalhes sobre os *tweets* de origem (as postagens iniciais) e suas informações associadas.

5.2 Preparação - Experimento evento único

Embora o conjunto de dados PHEME contenha cinco eventos distintos para a avaliação da detecção precoce de rumores, o tamanho limitado da amostra e a qualidade dos dados impõem desafios na simulação de situações do mundo real. Para enriquecer a pesquisa com cenários diferentes, dois experimentos foram projetados.

Para o primeiro experimento, apenas o evento *Charlie Hebdo* foi selecionado por ter um número razoável de publicações e interações para treinar algoritmos de aprendizado profundo. Outro motivo é o percentual de rumores que, embora seja alto comparado ao cenário real, é o mais baixo de todos os eventos.

Basicamente, o objetivo desse experimento é avaliar o desempenho dos algoritmos escolhidos em um conjunto de dados com postagens que tratam do mesmo contexto, nesse caso, todas as postagens se referem ao atentado terrorista que atingiu o jornal satírico francês *Charlie Hebdo* em 7 de janeiro de 2015, em Paris. Em um cenário de

uma rede social, esse experimento de um único contexto pode não fazer sentido, devido à diversidade de assuntos gerados. Porém, esse teste pode servir de prova de conceito para GNN's considerando a inexistência de um *corpus* mais robusto. Do ponto de vista prático, um algoritmo de detecção de rumor treinado e testado em um mesmo contexto, pode ter utilidade em fóruns menores ou em uma rede social onde é possível a filtragem de tópicos.

Com o conjunto de dados em formato de grafo e tabular já preparados, o experimento seguiu as etapas descritas na Figura 19.

1. **Sistema de Filtragem Baseado no Tempo**

   Esse sistema foi desenvolvido com base em dois requisitos principais, para se aproximar ao máximo do cenário real:

   - ❏ Primeiramente, conjunto de treinamento foi fixado em 70% do conjunto de dados total, selecionado em ordem cronológica para evitar vazamento de dados (ou seja, evitar usar postagens futuras para prever postagens passadas). Os 30% restantes foram reservados para o conjunto de teste.
   - ❏ Em sequência, foi desenvolvida uma classe em *Python* para realizar múltiplas inferências no conjunto de teste, de forma incremental, filtrando os dados em passos de 10 minutos, a partir do tempo que marca o início do conjunto de dados de teste, onde 70% dos posts já foram publicados. Durante cada passo, as métricas precisão, revocação, AUC, índice f1 e a quantidade de rumores em comparação a não-rumores foram registradas.
   - ❏ Por último, considerando que as postagens destinadas ao conjunto de treino podem ter interações a qualquer momento, antes de cada etapa de inferência, acontece o processo de re-treinamento do modelo, para computar eventuais atualizações nos atributos temporais das postagens do conjunto de treino, e por consequência, adicionar novas arestas no grafo de treino quando o algoritmo escolhido for uma rede neural de grafo.

2. **Comparação de Modelos**

   - ❏ Duas redes neurais em grafos foram avaliadas, especificamente a HAN *(Heterogeneous Graph Attention Network)* e GAT *(Graph Attention Network).*

- Além disso, modelos tradicionais de aprendizado de máquina conhecidos pelo bom desempenho na detecção de rumores foram testados, incluindo *Light Gbm, Random Forest* e LSTM (*Long Short-Term Memory*).
- Ao contrário das redes GNN, os modelos tradicionais utilizaram o formato de dados tabular, mas as condições de treinamento e teste foram as mesmas.

5.3 Resultados - Experimento Evento único

Considerando apenas o evento Charlie Hebdo, a Tabela 10 mostra a média das métricas AUC, precisão, *recall* e *f1* consideradas as mais importantes para este estudo. Observou-se que a revocação média foi significativamente melhor com os modelos LSTM e HAN.

Tabela 10 – Métricas por algoritmo - Evento *Charlie Hebdo*

Model	**AUC**	**Recall**	**Precisão**	**F1 Score**
Light GBM	**0.92**	0.59	0.64	0.62
HAN	0.91	0.76	**0.78**	**0.77**
Random Forest	0.89	0.71	0.59	0.64
GAT	0.88	0.71	0.76	0.73
LSTM	0.85	**0.8**	0.61	0.69

Como relação à precisão, as GNN's alcançaram um desempenho maior em comparação com os modelos do tipo "conjunto de classificadores"e a rede LSTM. Para visualizar os resultados ao longo do tempo e entender como as redes neurais de grafo apresentam um equilíbrio maior entre precisão e revocação, as Figuras 21 e 22 ilustram o comportamento dessas métricas desde o início até o fim do evento para todos os algoritmos testados.

Pode-se observar que as GNN's e LSTM apresentam variabilidade nas métricas, com destaque para a rede LSTM que aparenta não ter um comportamento estável. Porém, as GNN's permanecem acima de um limite aceitável de 75% durante a maior parte do tempo no gráfico de precisão e 70% no gráfico de revocação. Em contraste, os modelos do tipo "conjunto de classificadores" apresentam uma queda acentuada na precisão à medida que o número de novas postagens aumenta, levando a um aumento significativo nos falsos positivos. Esse comportamento também é observado na rede LSTM.

Ainda sobre a rede LSTM, foi observada um desempenho médio melhor que os algoritmos de *boosting*, especialmente se considerar a revocação como métrica mais importante,

com valor médio também maior que as GNN's. Porém, a precisão se deteriora muito ao longo do tempo, ficando abaixo de 60% em alguns momentos, e a sua alta variabilidade durante a inferência é um problema para o uso prático.

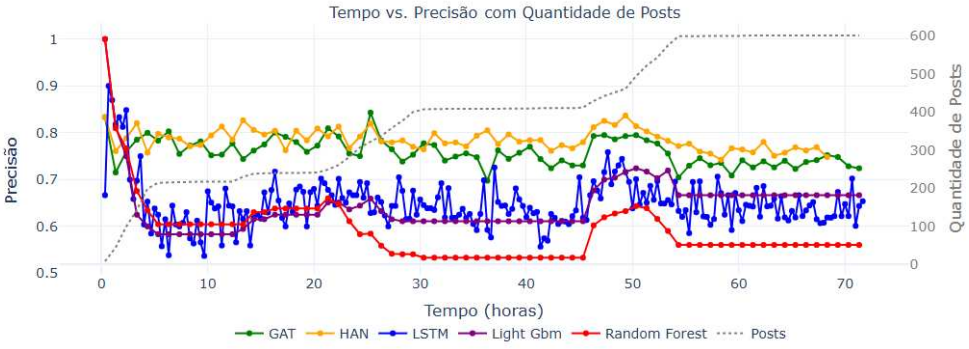

Figura 21 – Precisão e Revocação vs quantidade de posts acumulado por hora - HAN

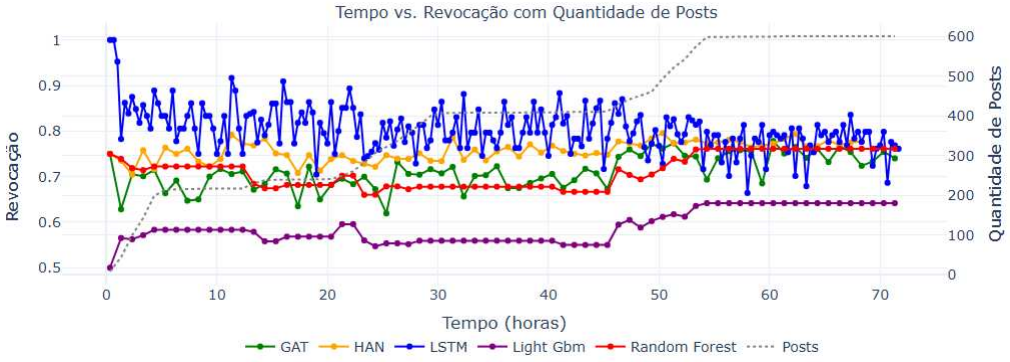

Figura 22 – Precisão e Revocação vs quantidade de posts acumulado por hora - GAT

5.4 Preparação – Experimento Transferência de aprendizado

Embora o primeiro experimento tenha permitido uma comparação entre o desempenho de redes neurais de grafos e modelos tradicionais, o uso do dado de um único evento, em que todos os *tweets* são de um mesmo contexto limita o uso completo do

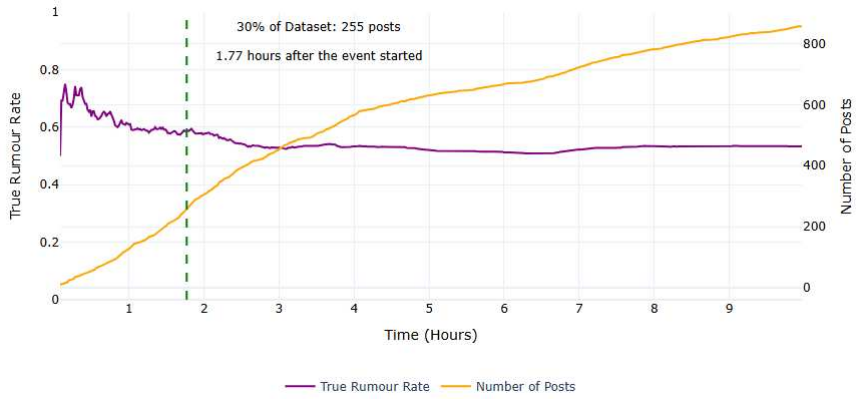

Figura 23 – Tempo vs. Taxa de rumores verdadeiros e quantidade de publicações

conjunto de dados *PHEME*. Além disso, é importante verificar se as redes neurais de grafos são capazes de ter um bom desempenho de detecção, mesmo sendo treinadas e testadas em eventos diferentes. Vale lembrar que o conteúdo textual irá divergir de um evento para outro; porém, a premissa é que o padrão da propagação de rumor processado pelas GNN's ajude a superar o desempenho dos algoritmos tabulares que são mais dependentes do conteúdo textual.

Para preencher essa lacuna, foi proposta uma abordagem de aprendizado por transferência, permitindo que os modelos fossem treinados em um evento específico e avaliados em outros, com contextos diferentes. Essa abordagem simula um cenário em que um conjunto de dados previamente coletado é utilizado para detectar boatos em eventos futuros.

O processo descrito para o experimento com um único evento permanece o mesmo; no entanto, durante a fase de treinamento, 100% dos dados do evento *Charlie Hebdo* foram utilizados, juntamente com os 30% iniciais dos dados do evento de teste (selecionados em ordem cronológica), para ajustar o modelo previamente treinado com os dados do evento de origem. Como resultado, o sistema de filtragem baseado no tempo opera da mesma forma no conjunto de teste, garantindo que o modelo seja avaliado em publicações não vistas durante o treinamento.

Para exemplificar melhor, a Figura 23 mostra em que ponto do conjunto de dados acontece o corte para o evento *Germanwings*, mostrando a quantidade de publicações e a taxa de rumores verdadeiros no tempo.

Para cada um dos eventos a quantidade de publicações e a taxa de rumores verdadeiros no corte de 30% foi:

- *Ferguson*: 8.3% e 181 publicações
- *Germanwings*: 42,7% e 297 publicações
- *Ottawa*: 52,5% e 453 publicações
- *Sydney*: 56,4% e 421 publicações

Essa informação da taxa de rumores verdadeiros no corte de 30% referente ao conjunto de dados de treino de ajuste concatenado com todos os dados do evento *Charlie Hebdo* será de fundamental importância para entender o desempenho dos algoritmos em cada evento.

5.5 Resultados - Transferência de aprendizado

Os resultados para todos os eventos e algoritmos estão resumidos na Tabela 11. Um padrão semelhante foi observado no experimento de transferência de aprendizado em relação ao experimento com evento único. As GNN's demonstraram um bom equilíbrio entre precisão e revocação. Para ilustrar o comportamento temporal de todos os algoritmos, as Figuras 24 a 27 mostram a revocação e precisão ao longo do tempo para todos os algoritmos e conjunto de dados testados.

Tabela 11 – Métricas de desempenho - Transferência de aprendizado

Modelo	**Dataset**	**AUC (média)**	**Recall (média)**	**Precisão (média)**	**F1-score (média)**
HAN	*Ferguson*	**0.84**	**0.64**	**0.75**	**0.69**
GAT	*Ferguson*	0.81	0.57	0.73	0.64
Light Gbm	*Ferguson*	0.71	0.08	0.26	0.14
Random Forest	*Ferguson*	0.73	0.32	0.50	0.37
LSTM	*Ferguson*	0.51	0.31	0.26	0.23
HAN	*Germanwings*	0.80	0.63	0.63	0.63
GAT	*Germanwings*	**0.83**	0.66	**0.66**	**0.66**
Light Gbm	*Germanwings*	0.71	0.55	0.48	0.51
Random Forest	*Germanwings*	0.65	0.57	0.45	0.50
LSTM	*Germanwings*	0.8	**0.9**	0.48	0.62
HAN	*Ottawa*	0.86	**0.79**	0.80	**0.80**
GAT	*Ottawa*	0.84	0.76	0.77	0.76
Light Gbm	*Ottawa*	**0.87**	0.52	**0.87**	0.65
Random Forest	*Ottawa*	0.85	0.60	0.84	0.70
LSTM	*Ottawa*	0.8	0.74	0.75	0.75
HAN	*Sydney*	0.81	**0.76**	**0.72**	**0.74**
GAT	*Sydney*	0.80	0.75	0.71	0.73
Light Gbm	*Sydney*	**0.85**	0.64	**0.72**	0.68
Random Forest	*Sydney*	0.84	0.67	0.71	0.69
LSTM	*Sydney*	0.77	0.67	0.66	0.66

A inicialização e comportamento das GNN's e os demais algoritmos foram bem diferentes em cada evento.

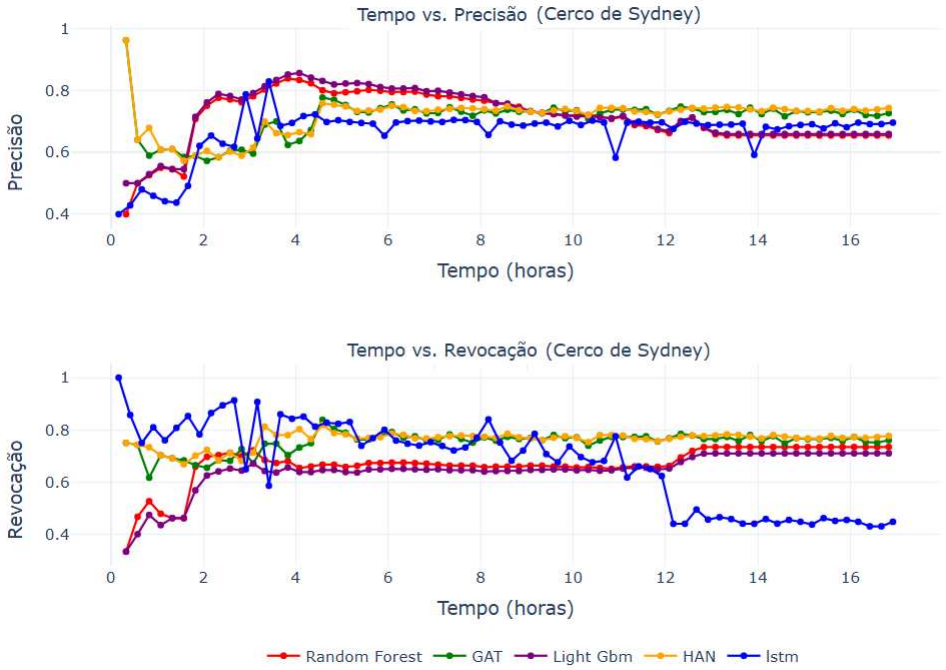

Figura 24 – Precisão e Revocação por hora e algoritmo - Sydney Siege

1. *Sydney Siege*: Precisão e revocação inicial baixo para os algoritmos *Random Forest* e *Light Gbm* e mais alto para as GNN's. A estabilização das duas métricas se deu por volta da hora 5 desde o início do evento, em que as GNN's foram superadas em precisão até a hora 8 horas do evento. A revocação quase sempre foi maior para as GNN's, sendo algumas vezes superadas pela rede LSTM que apresentou um comportamento instável.

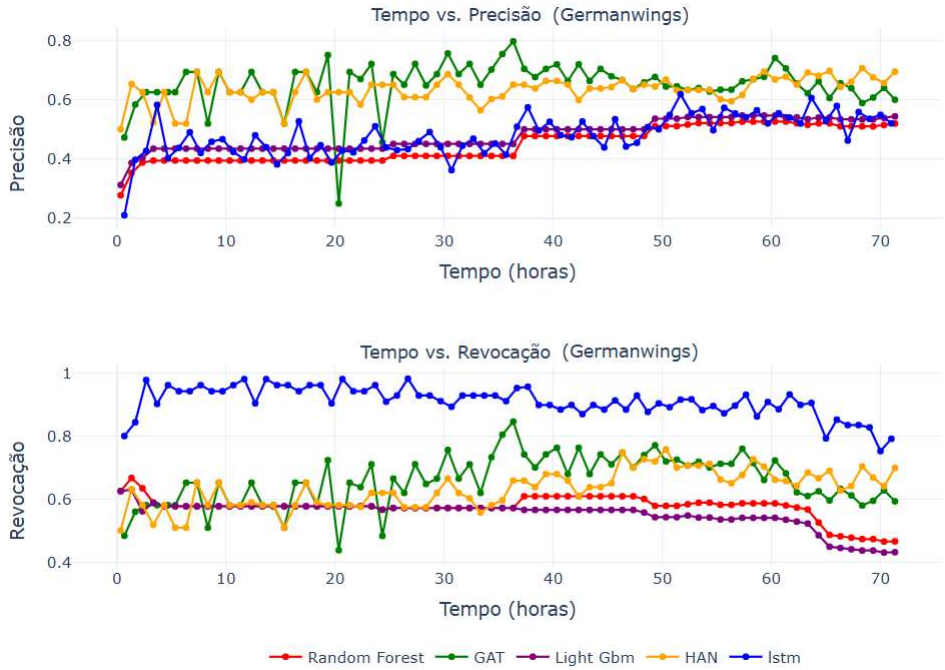

Figura 25 – Precisão e Revocação por hora e algoritmo - *Germanwings Crash*

2. *Germanwings Crash*: Evento com duração longa, onde as GNN's mostraram uma variabilidade alta com o surgimento de novos *posts* e interações. Porém, mesmo com essa oscilação, em poucos pontos de inferência houve desempenho menor que os algoritmos *Random Forest* e *Light Gbm* , os quais apresentaram uma queda brusca na revocação ao final do evento. A rede LSTM apresentou um comportamento ainda instável, com uma revocação alta, mas ao custo de uma precisão média baixa.

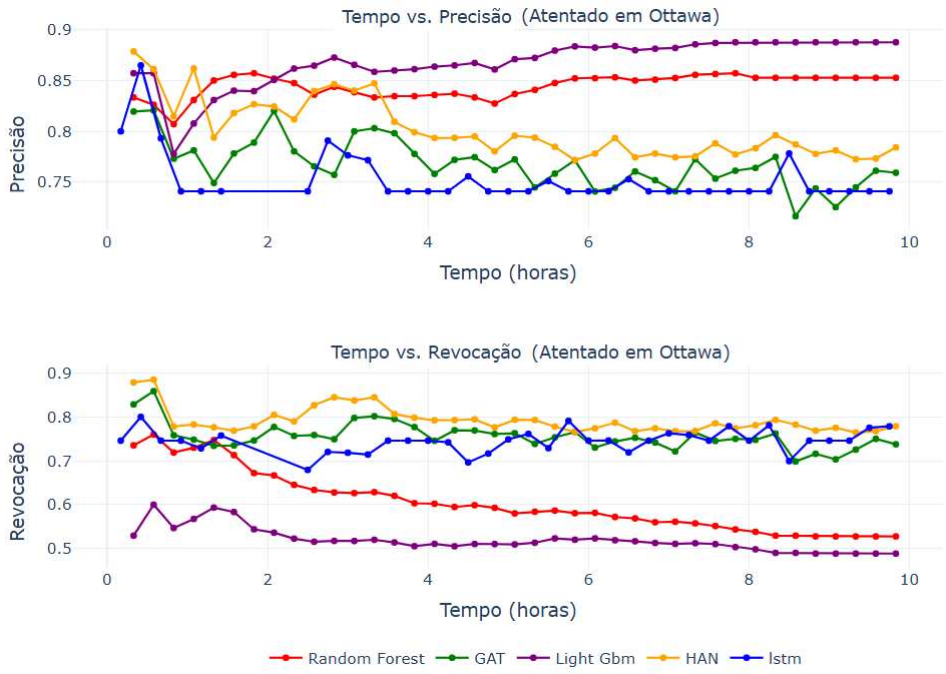

Figura 26 – Precisão e Revocação por hora e algoritmo - *Ottawa Shooting*

3. *Ottawa Shooting*: Neste evento, os algoritmos *Random Forest* e *Light Gbm* tiveram uma boa precisão, gerando poucos falsos positivos durante todo o evento, porém a revocação que começou acima de 70% foi decrescendo até quase 50% ao final do evento. Já as GNN's tiveram uma precisão menor, mas ainda entre 75% e 80% com uma revocação na mesma faixa, mostrando um desempenho equilibrado. A rede LSTM apresentou uma revocação estável, porem uma precisão mais baixa de todos os algoritmos, com uma queda brusca na hora 1 do evento.

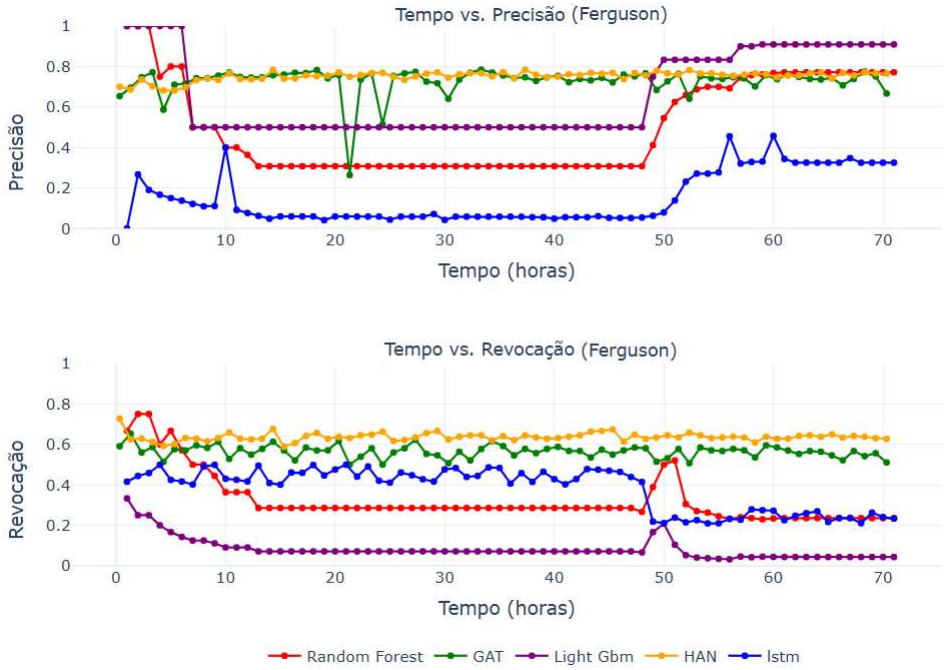

Figura 27 – Precisão e Revocação por hora e algoritmo - *Ferguson*

4. *Ferguson*: Este foi o evento mais desafiador, considerando o baixo número de rumores verdadeiros no conjunto de dados de treino de ajuste. Mais uma vez as GNN's tiveram um desempenho equilibrado, sem apresentar a mesma queda brusca como os algoritmos de conjunto após a hora 10. Em exceção as GNN's, todos os outros algoritmos geraram uma quantidade significativa de falsos positivos, mostrando que o treino não foi eficiente, mesmo com o controle de desbalanceamento de classes.

5.6 Considerações Finais

Conforme os resultados apresentados na Tabela 11 e observando as Figuras 24 a 27 foi gerado um *ranking* de desempenho dos algoritmos e qual recomendação de aplicação prática de cada um.

1. *Heterogeneous Graph Attention Network* (HAN) : As GNN's tiveram a melhor precisão e revocação em todos os eventos, como consequência o melhor índice f1. Também tiveram a melhor AUC em dois dos quatro eventos. Em comparação com

a outra rede testada, a GAT, a HAN foi um pouco superior em três dos quatro eventos, sendo assim, fica em primeiro no ranking de desempenho.

2. Graph Attention Network (GAT): A GAT teve o melhor desempenho no evento *Germanwings* e apresentou métricas consistentes em todos os demais eventos. Assim, fica em segundo.

3. LSTM: Apesar da instabilidade, a rede LSTM mostrou um desempenho ligeiramente melhor nos eventos do tiroteio de *Ottawa* e no *Germanwings*. Isso mostra que com a otimização de parâmetros da rede pode se conseguir métricas melhores. Porém, há um custo computacional mais elevado do que os algoritmos *Light Gbm* e Random forest.

4. *Light Gbm*: Apresentou a melhor precisão em 2 dos quatro eventos. Como os demais algoritmos tabulares, teve um péssimo desempenho no evento *Ferguson*. Sendo o desempenho similar ao Random Forest, do ponto de vista prático prefere-se o *Light Gbm* devido a sua menor complexidade de treinamento.

5. Random Forest: Esse algoritmo fica em último lugar apenas pela praticidade do *Light Gbm*, sendo uma boa alternativa para modelos de base em problemas de detecção de rumor.

A Tabela resume por algoritmo a quantidade de vezes que foi melhor em cada métrica.

Tabela 12 – Resumo - Superioridade por algoritmo em cada métrica

Modelo	**AUC**	**Precisão**	**Recall**	**F1-score**
HAN	1	3	3	3
GAT	1	1	1	1
Light GBM	1	1	0	0
Random Forest	1	0	0	0
LSTM	0	0	0	0

Capítulo 6

Conclusão

Por meio do desenvolvimento dessa dissertação, foi possível conhecer o cenário atual da área de pesquisa em algoritmos e técnicas de detecção de rumor. Notou-se uma evolução do tema a partir do surgimento das redes sociais, com as primeiras abordagens associadas a algoritmos supervisionados simples passando ao aprendizado profundo até a área das redes neurais de grafos que são o objetivo de investigação deste trabalho. A partir da seleção de duas redes neurais de grafos e outros três algoritmos com histórico de bom desempenho na tarefa de detecção de rumor *(LSTM, Random Forest e Light GBM)*, foi necessário criar um método padronizado para comparação justa e monitoramento contínuo das métricas de classificação durante intervalos de tempo a cada 10 minutos do início ao fim do evento. Esse método, referido no trabalho com **ERDB**, demonstrou uma boa capacidade de adaptação para diferentes tipos de algoritmos e representações de dados, do tipo tabular e de grafo. Além disso, a aplicação da divisão cronológica de treino e teste preveniu efetivamente vazamentos de dados e permitiu o retreinamento dos algoritmos conforme novas interações surgiram para as publicações no conjunto de dados de treino. Com relação ao desempenho dos algoritmos, os resultados foram similares nos experimentos de evento único e de transferência de aprendizado, em que as redes neurais de grafo apresentaram métricas mais estáveis, com um bom equilíbrio entre precisão e revocação. O segundo algoritmo com mais potencial foi a rede LSTM, que, embora tenha mostrado um comportamento estável, apresentou uma revocação e precisão média mais alta que os classificadores *Light GBM* e *Random Forest*, contribuindo para um índice f1 mais alto. Os gráficos temporais de precisão e revocação mostraram que as GNN's apresentam uma variabilidade constante dessas métricas, pela natureza do seu processo de treinamento que é mais instável, porém, se mantêm acima de um patamar

aceitável ao longo do tempo. A rede LSTM também se comporta da mesma forma, porém com valores de revocação que podem ser maiores e menores de precisão. Os algoritmos *Random Forest* e *Light GBM* apresentaram pouca variabilidade nas métricas; porém, nos eventos *Ottawa Shooting*, *Germanwings Crash* e *Ferguson*, a capacidade de prever rumores verdadeiros foi prejudicada ao final da série temporal. Ficou claro que os mais desafiadores foram aqueles que tiveram um desbalanceamento de classe alto no conjunto de dados de treino, feita a divisão nas 30% primeiras horas de cada evento para o experimento de transferência de aprendizado.

6.1 Principais Contribuições

Como contribuição para o cenário prático, o método *ERDB* se mostrou útil na comparação rápida e intercambiável para teste de diferentes algoritmos, o que ajuda à tomada de decisão com base em experimentos padronizados próximos ao cenário real, focados na detecção precoce.

Com relação às redes neurais de grafo, por meio do conjunto de dados PHEME e a padronização dos testes, foi possível observar que, através do processamento da informação da propagação, pode-se conseguir um classificador de rumor mais homogêneo, com equilíbrio entre precisão e revocação e baixa degradação dessas métricas ao longo do tempo. Também ficou evidente que há outras alternativas, como a rede LSTM que podem ser mais eficientes quando a necessidade é detectar a maior quantidade de rumores possível, mesmo com o custo de muitos falsos positivos.

No cenário acadêmico, o método utilizado e todo código produzido poderão servir de ferramenta de testes rápidos para novas técnicas de detecção de rumor, pois permitirá a comparação rápida com algoritmos básicos da literatura e poderá variar a utilização de dados tabulares e de grafo. Todo código gerado nesse trabalho será disponibilizado abertamente em https://github.com/douglas-dm9/rumour-detection-gnn.

6.2 Trabalhos Futuros

Para continuar o desenvolvimento do tema, há vários caminhos que podem ser trilhados, considerando a riqueza e potencial desta área, como, por exemplo:

- Utilizar conjunto de dados maiores para avaliar o desempenho de redes neurais de grafos, especialmente voltados à língua portuguesa. Há uma carência enorme na

obtenção de bases confiáveis e o trabalho de rotulação do que é rumor/não rumor é pesado, mas pode ser reduzido com a introdução dos modelos de linguagem natural generativa e aprendizado supervisionado.

- ❏ Na aplicação de redes neurais de grafos, recomenda-se testar diferentes configurações de grafos como, por exemplo: diferentes tipos de nós e arestas, incluindo mais de um tipo de interações, arestas com pesos e toda informação que possa ser adiciona para extrair o máximo de potencial das GNN's que processam grafos heterogêneos.

- ❏ No uso de informação textual, há várias oportunidades de melhoria pela investigação de qual algoritmo de geração de *embeddings* tem melhor desempenho para a detecção de rumor. Nesse trabalho, optou-se por utilizar um modelo pré-treinado de uso gratuito e baixo custo computacional, porém modelos pagos via *API* ou gratuitos mais robustos podem capturar melhor a informação textual para a detecção de rumor.

- ❏ Promover o comparativo de diferentes algoritmos na tarefa de detecção precoce de rumor, considerando a melhor versão de cada um, através da otimização de hiper parâmetros, o que não foi possível neste trabalho por limitação de tempo e recursos computacionais. Algoritmos como *Light GBM* e *Random Forest* podem melhorar significativamente de desempenho através da escolha de parâmetros apropriados.

6.3 Contribuições em Produção Bibliográfica

O artigo *Early rumour detection: Evaluating the effectiveness of graph neural networks* foi submetido ao evento *12th IEEE International Conference on Data Science and Advanced Analytics (DSAA)* com o resumo da proposta e principais resultados alcançados nesta dissertação.

Referências

AGUIRRE, J. D. et al. Identificação e classificação de corpos d'água em imagem digital pela aplicação do método random forest. Universidade Federal do Pampa, 2023. Disponível em: <https://doi.org/10.1007/s13369-020-04839-2>.

ALSAEEDI, A.; AL-SAREM, M. Detecting rumors on social media based on a cnn deep learning technique. **Arab J Sci Eng**, v. 45, n. 12, p. 10813–10844, 2020.

ALZAMZAMI, F.; HODA, M.; SADDIK, A. E. Light gradient boosting machine for general sentiment classification on short texts: a comparative evaluation. **IEEE access**, IEEE, v. 8, p. 101840–101858, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2020.2997330>.

ASGHAR, M. Z. et al. Exploring deep neural networks for rumor detection. **J Ambient Intell Humaniz Comput**, v. 12, n. 4, p. 4315–4333, 2021. Disponível em: <https://doi.org/10.1007/s12652-019-01527-4>.

AZRI, A. et al. Calling to cnn-lstm for rumor detection: A deep multi-channel model for message veracity classification in microblogs. Springer, v. 12979, p. 497–513, 2021. Disponível em: <https://doi.org/10.1007/978-3-030-86517-7_31>.

BAI, N. et al. Rumour detection based on graph convolutional neural net. **IEEE Access**, v. 9, p. 21686–21693, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3050563>.

BERG, R. v. d.; KIPF, T. N.; WELLING, M. Graph convolutional matrix completion. **arXiv preprint arXiv:1706.02263**, 2017.

BERNSTEIN, M. **Graph convolutional neural networks**. 2023. Accessed: 2025-05-10. Disponível em: <https://mbernste.github.io/posts/gcn/>.

BIAN, T. et al. Rumor detection on social media with bi directional graph convolutional networks. In: **Proceedings of the 34th AAAI Conference on**

Artificial Intelligence (AAAI 2020). [s.n.], 2020. p. 549–556. Disponível em: <https://doi.org/10.1609/aaai.v34i01.5393>.

BISPO, T. D. Arquitetura lstm para classificação de discursos de ódio cross-lingual inglês-ptbr. Pós-Graduação em Ciência da Computação, 2018.

Brasil. Secretaria de Comunicação Social. **Inscrições do CPNU 2025 não estão abertas**. 2025. <https://www.gov.br/secom/pt-br/fatos/brasil-contra-fake/noticias/2025/03/inscricoes-do-cnpu-2025-nao-estao-abertas>. Acesso em: 19 abr. 2025.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001. Disponível em: <https://doi.org/10.1023/A:1010933404324>.

BROADWATER, K.; STILLMAN, N. **Graph Neural Networks in Action**. [S.l.]: Manning, 2025. (In Action). ISBN 9781617299056.

CASTILLO, C.; MENDOZA, M.; POBLETE, B. Information credibility on twitter. In: **Proceedings of the 20th international conference on World wide web**. [s.n.], 2011. p. 675–684. Disponível em: <https://doi.org/10.1145/1963405.1963500>.

CEN, J.; LI, Y. A rumor detection method from social network based on deep learning in big data environment. **Comput Intell Neurosci**, 2022. Disponível em: <https://doi.org/10.1155/2022/1354233>.

CEN, J.; LI, Y. et al. A rumor detection method from social network based on deep learning in big data environment. **Computational Intelligence and Neuroscience**, Hindawi, v. 2022, 2022. Disponível em: <https://doi.org/10.1155/2022/1354233>.

CHEN, T. et al. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. In: **Trends and Applications in Knowledge Discovery and Data Mining - PAKDD 2018 Workshops**. Springer, 2018, (Lecture Notes in Computer Science, v. 11154). p. 40–52. Disponível em: <https://doi.org/10.1007/978-3-030-04503-6_4>.

FEY, M. **Heterogeneous Graphs**. 2024. <https://pytorch-geometric.readthedocs.io/en/2.5.1/notes/heterogeneous.html>. Accessed: 2025-07-21. Disponível em: <https://pytorch-geometric.readthedocs.io/en/2.5.1/notes/heterogeneous.html>.

FEY, M.; LENSSEN, J. E. Fast graph representation learning with pytorch geometric. In: **ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds**. [s.n.], 2019. Disponível em: <https://arxiv.org/abs/1903.02428>.

FRASCONI, P.; GORI, M.; SPERDUTI, A. A general framework for adaptive processing of data structures. **IEEE transactions on Neural Networks**, IEEE, v. 9, n. 5, p. 768–786, 1998. Disponível em: <https://doi.org/10.1109/72.712151>.

GALLICCHIO, C.; MICHELI, A. Fast and deep graph neural networks. In: **Proceedings of the AAAI conference on artificial intelligence**. [S.l.: s.n.], 2020. v. 34, n. 04, p. 3898–3905.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. In: ACM. **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. 2016. p. 855–864. Disponível em: <https://doi.org/10.1145/2939672.2939754>.

GUO, M. et al. An adaptive deep transfer learning model for rumor detection without sufficient identified rumors. **Math Probl Eng**, 2020. Disponível em: <https://doi.org/10.1155/2020/7562567>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HUANG, Q. et al. Heterogeneous graph attention networks for early detection of rumors on twitter. In: IEEE. **2020 international joint conference on neural networks (IJCNN)**. 2020. p. 1–8. Disponível em: <https://doi.org/10.1109/IJCNN48605.2020.9207582>.

ISLAM, M. R. et al. Deep learning for misinformation detection on online social networks: a survey and new perspectives. **Social Network Analysis and Mining**, Springer, v. 10, n. 1, p. 82, 2020. Disponível em: <https://doi.org/10.1007/s13278-020-00696-x>.

JAFFAR, H.; MOHADES, A.; ABADY, M. E. S. A. Stylegraph: A heterogeneous graph neural framework for stylistic and semantic rumor detection on social media. **IEEE Access**, IEEE, 2025. Disponível em: <https://doi.org/10.1109/ACCESS.2025.3595498>.

KE, G. et al. Lightgbm: A highly efficient gradient boosting decision tree. **Advances in neural information processing systems**, v. 30, 2017.

KHATUA, A. et al. Igb: Addressing the gaps in labeling, features, heterogeneity, and size of public graph datasets for deep learning research. In: **Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining**. [s.n.], 2023. p. 4284–4295. Disponível em: <https://doi.org/10.1145/3580305.3599843>.

KOCHKINA, E.; LIAKATA, M.; ZUBIAGA, A. Pheme dataset for rumor detection and veracity classification. In: **Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)**. [s.n.], 2018. p. 1–8. Disponível em: <https://doi.org/10.1145/3161603>.

KOTTETI, C. M. M.; DONG, X.; QIAN, L. Ensemble deep learning on time-series representation of tweets for rumor detection in social media. **CoRR**, 2020. Disponível em: <https://doi.org/10.3390/app10217541>.

KREBS, V. **Political Books Network (2004)**. 2004. <https://networks.skewed.de/net/polbooks>. Accessed: 2025-05-03.

KUMAR, A.; BHATIA, M. P. S.; SANGWAN, S. R. Rumour detection using deep learning and filter-wrapper feature selection in benchmark twitter dataset. **Multim Tools Appl**, v. 81, n. 24, p. 34615–34632, 2022. Disponível em: <https://doi.org/10.1007/s11042-021-11340-x>.

KWON, S. et al. Prominent features of rumor propagation in online social media. In: IEEE. **2013 IEEE 13th international conference on data mining**. 2013. p. 1103–1108. Disponível em: <https://doi.org/10.1109/ICDM.2013.61>.

LABONNE, M. **Hands-On Graph Neural Networks Using Python: Practical techniques and architectures for building powerful graph and deep learning apps with PyTorch**. [S.l.]: Packt Publishing Ltd, 2023.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998. Disponível em: <https://doi.org/10.1109/5.726791>.

LI, Q.; ZHANG, Q.; SI, L. Rumor detection by exploiting user credibility information, attention and multi-task learning. In: **Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019**. Association for Computational Linguistics, 2019. p. 1173–1179. Disponível em: <https://doi.org/10.18653/v1/P19-1113>.

LIU, B. et al. Nowhere to hide: Online rumor detection based on retweeting graph neural networks. **IEEE transactions on neural networks and learning systems**, IEEE, 2022.

LIU, Y.; JIN, X.; SHEN, H. Towards early identification of online rumors based on long short-term memory networks. **Inf Process Manag**, v. 56, n. 4, p. 1457–1467, 2019. Disponível em: <https://doi.org/10.1016/j.ipm.2018.11.003>.

MA, J.; GAO, W. Detecting rumors from microblogs with recurrent neural networks. In: **Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)**. [S.l.: s.n.], 2016. p. 3818–3824.

MA, J.; GAO, W.; MITRA, P. Detecting rumors from microblogs with recurrent neural networks. AAAI Press, 2016.

MA, J. et al. Detect rumors using time series of social context information on microblogging websites. In: **Proceedings of the 24th ACM international on**

conference on information and knowledge management. [s.n.], 2015. p. 1751–1754. Disponível em: <https://doi.org/10.1145/2806416.2806607>.

MA, J.; GAO, W.; WONG, K.-F. Detect rumors in microblog posts using propagation structure via kernel learning. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. 2017. Disponível em: <https://doi.org/10.18653/v1/P17-1066>.

MATHIEU, M.; COUPRIE, C.; LECUN, Y. Deep multi-scale video prediction beyond mean square error. **arXiv preprint arXiv:1511.05440**, 2015.

MONTIEL, J. et al. River: machine learning for streaming data in python. **Journal of Machine Learning Research**, v. 22, p. 1–?, ??, 2021. Open-source library available at <https://github.com/online-ml/river>.

OLAH, C. **Understanding LSTM Networks**. 2015. Accessed: 2025-05-03. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

PATTANAIK, B.; MANDAL, S.; TRIPATHY, R. A survey on rumor detection and prevention in social media using deep learning. **Knowledge and Information Systems**, v. 65, n. 10, p. 3839–3880, 2023. Disponível em: <https://doi.org/10.1007/s10115-023-01902-w>.

PATTANAIK, B.; MANDAL, S.; TRIPATHY, R. M. A survey on rumor detection and prevention in social media using deep learning. **Knowledge and Information Systems**, Springer, p. 1–42, 2023. Disponível em: <https://doi.org/10.1007/s10115-023-01902-w>.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [s.n.], 2014. p. 1532–1543. Disponível em: <https://doi.org/10.3115/v1/D14-1162>.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. In: ACM. **Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. 2014. p. 701–710. Disponível em: <https://doi.org/10.1145/2623330.2623732>.

POTTHAST, M. et al. A stylometric inquiry into hyperpartisan and fake news. **arXiv preprint arXiv:1702.05638**, 2017. Disponível em: <https://doi.org/10.18653/v1/P18-1022>.

RUBIN, V. L. On deception and deception detection: Content analysis of computer-mediated stated beliefs. **Proceedings of the American Society for Information Science and Technology**, Wiley Online Library, v. 47, n. 1, p. 1–10, 2010. Disponível em: <https://doi.org/10.1002/meet.14504701124>.

SONG, C. et al. Ced: Credible early detection of social media rumors. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 33, n. 8, p. 3035–3047, 2019. Disponível em: <https://doi.org/10.1109/TKDE.2019.2961675>.

SUJANA, Y.; LI, J.; KAO, H. Rumor detection on twitter using multiloss hierarchical bilstm with an attenuation factor. In: **Proceedings of AACL/IJCNLP 2020**. Association for Computational Linguistics, 2020. p. 18–26. Disponível em: <https://doi.org/10.18653/v1/2020.aacl-main.3>.

TAN, L. et al. Research status of deep learning methods for rumor detection. **Multimedia Tools and Applications**, Springer, v. 82, n. 2, p. 2941–2982, 2023. Disponível em: <https://doi.org/10.1007/s11042-022-12800-8>.

TU, K.; CHEN, C.; HOU, C. Rumor2vec: A rumor detection framework with joint text and propagation structure representation learning. **Inf Sci**, v. 560, p. 137–151, 2021. Disponível em: <https://doi.org/10.1016/j.ins.2020.12.080>.

TU, K. et al. Rumor2vec: a rumor detection framework with joint text and propagation structure representation learning. **Information Sciences**, Elsevier, v. 560, p. 137–151, 2021.

VASWANI, A. et al. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

VELIČKOVIĆ, P. et al. Graph attention networks. **arXiv preprint arXiv:1710.10903**, 2017.

WANG, G. et al. Region-enhanced deep graph convolutional networks for rumor detection. **CoRR**, 2022.

WANG, S. K. X.; RAJTMAJER, S. The failed migration of academic twitter. **Pennsylvania State University**, June 2025. Disponível em: <https://arxiv.org/pdf/2406.04005v1>.

WANG, W. Y. "liar, liar pants on fire": A new benchmark dataset for fake news detection. **arXiv preprint arXiv:1705.00648**, 2017. Disponível em: <https://doi.org/10.18653/v1/P17-2067>.

WANG, X. et al. Heterogeneous graph attention network. In: **The world wide web conference**. [s.n.], 2019. p. 2022–2032. Disponível em: <https://doi.org/10.1145/3308558.3313562>.

WU, K.; YANG, S.; ZHU, K. Q. False rumors detection on sina weibo by propagation structures. In: IEEE. **2015 IEEE 31st international conference on data engineering**. 2015. p. 651–662. Disponível em: <https://doi.org/10.1109/ICDE.2015.7113322>.

WU, L. et al. Misinformation in social media: definition, manipulation, and detection. **ACM SIGKDD explorations newsletter**, ACM New York, NY, USA, v. 21, n. 2, p. 80–90, 2019.

WU, Y. et al. Dual-channel early rumor detection based on factual evidence. **Expert Systems with Applications**, Elsevier, v. 238, p. 121928, 2024. Disponível em: <https://doi.org/10.1016/j.eswa.2023.121928>.

YANG, Z. et al. A model for early rumor detection base on topic-derived domain compensation and multi-user association. **Expert Systems with Applications**, Elsevier, v. 250, p. 123951, 2024. Disponível em: <https://doi.org/10.1016/j.eswa.2024.123951>.

YU, D.; ZHOU, Y. et al. Heterogeneous graph convolutional network-based dynamic rumor detection on social media. **Complexity**, Hindawi, v. 2022, 2022.

YU, D. et al. Heterogeneous graph convolutional network-based dynamic rumor detection on social media. **CompLex**, p. 8393736:1–8393736:10, 2022.

YU, F. et al. A convolutional approach for misinformation identification. In: **IJCAI**. [s.n.], 2017. p. 3901–3907. Disponível em: <https://doi.org/10.24963/ijcai.2017/545>.

YUAN, C.; MA, Q.; ZHOU, W. Early detection of fake news by utilizing the credibility of news, publishers, and users based on weakly supervised learning. **arXiv preprint arXiv:2012.04233**, 2020. Disponível em: <https://doi.org/10.1109/ICDM.2019.00090>.

YUAN, C. et al. Jointly embedding the local and global relations of heterogeneous graph for rumor detection. In: IEEE. **2019 IEEE international conference on data mining (ICDM)**. 2019. p. 796–805. Disponível em: <https://doi.org/10.1109/ICDM.2019.00090>.

ZAHARIA, M. et al. Accelerating the machine learning lifecycle with MLflow. In: **Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**. [S.l.: s.n.], 2018. Open-source: <https://mlflow.org>.

ZHOU, J. et al. Graph neural networks: A review of methods and applications. **AI open**, Elsevier, v. 1, p. 57–81, 2020. Disponível em: <https://doi.org/10.1016/j.aiopen.2021.01.001>.

ZHOU, K. et al. Early rumour detection. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. [S.l.: s.n.], 2019. p. 1614–1623.

ZHU, W. C. et al. Asymmetric transitivity preserving graph embedding. In: ACM. **Proceedings of the 22nd ACM SIGKDD International Conference on**

Knowledge Discovery and Data Mining. 2016. p. 1105–1114. Disponível em: <https://doi.org/10.1145/2939672.2939751>.

ZUBIAGA, A. et al. Detection and resolution of rumours in social media: A survey. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 2, p. 1–36, 2018.

ZUBIAGA, A.; LIAKATA, M.; PROCTER. Analysing how people orient to and spread rumours in social media by looking at conversational threads. **PLOS ONE**, Public Library of Science (PLoS), v. 11, n. 3, p. e0150989, mar. 2016. ISSN 1932-6203. Disponível em: <http://dx.doi.org/10.1371/journal.pone.0150989>.