unesp

UNIVERSIDADE ESTADUAL PAULISTA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

INTEGRAÇÃO DE LARGE LANGUAGE MODELS E GRAPH NEURAL NETWORKS PARA OTIMIZAR CLASSIFICAÇÃO DE TEXTO E EXTRAÇÃO DE CONHECIMENTO

LARISSA PESSOA SÁ MENEZES

INSTITUTO DE GEOCIÊNCIAS E CIÊNCIAS EXATAS

RIO CLARO

UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Instituto de Geociências e Ciências Exatas
Câmpus de Rio Claro

Larissa Pessoa Sá Menezes

Integração de Large Language Models e Graph Neural Networks para Otimizar Classificação de Texto e Extração de Conhecimento

Dissertação de Mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Câmpus de Rio Claro, da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Fabricio Aparecido Breve

Rio Claro-SP

2026

M543i

Menezes, Larissa Pessoa Sá
Integração de large language models e graph neural networks para otimizar classificação de texto e extração de conhecimento / Larissa Pessoa Sá Menezes. -- Rio Claro, 2026
69 p. : tabs., fotos

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Instituto de Geociências e Ciências Exatas, Rio Claro
Orientador: Fabrício Aparecido Breve

1. Inteligência artificial. 2. Redes neurais (Computação). 3. Processamento humano da informação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

IMPACTO POTENCIAL DESTA PESQUISA

Esta pesquisa contribui para o avanço de técnicas de inteligência artificial ao integrar modelos de linguagem de grande escala e redes neurais em grafos, possibilitando classificação textual mais precisa e extração automatizada de conhecimento, com potencial aplicação em áreas como saúde, educação e gestão da informação.

POTENTIAL IMPACT OF THIS RESEARCH

This research advances artificial intelligence techniques by integrating large language models and graph neural networks, enabling more accurate text classification and automated knowledge extraction, with potential applications in areas such as healthcare, education, and information management.

UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Instituto de Geociências e Ciências Exatas
Câmpus de Rio Claro

LARISSA PESSOA SÁ MENEZES

INTEGRAÇÃO DE LARGE LANGUAGE MODELS E GRAPH NEURAL NETWORKS PARA OTIMIZAR CLASSIFICAÇÃO DE TEXTO E EXTRAÇÃO DE CONHECIMENTO

Dissertação de Mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Câmpus de Rio Claro, da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Comissão Examinadora

Prof. Dr. FABRICIO APARECIDO BREVE (Orientador)
IGCE / UNESP / Rio Claro (SP)

Profa. Dra. SOLANGE OLIVEIRA REZENDE
ICMC / USP / São Carlos (SP)

Profa. Dra. VERONICA OLIVEIRA DE CARVALHO
IGCE / UNESP / Rio Claro (SP)

Conceito: Aprovado.

Rio Claro (SP), 13 de fevereiro de 2026.

Agradecimentos

Gostaria de agradecer imensamente ao Prof. Dr. Fabrício Breve pela orientação, paciência e parceria ao longo deste período. Sua condução, acompanhamento e recomendações técnicas foram fundamentais para a execução e qualidade deste trabalho.

De igual importância agradeço o apoio de minha família, que não é definida por laços de sangue mas através de vínculos genuínos de afeto, confiança e amor. Aos meus amigos e, especialmente, à minha companheira Ana Ximenes, obrigado por todo o suporte emocional e por compreenderem minha ausência nos momentos necessários para a elaboração desta dissertação.

Resumo

A representação estruturada do conhecimento humano e o avanço dos Large Language Models (LLMs) têm impulsionado novas fronteiras na Inteligência Artificial. Neste contexto, os grafos de conhecimento (KGs) emergem como ferramentas essenciais para organizar informações complexas, permitindo sua integração com técnicas de aprendizado profundo. Este trabalho introduz o GenGraph, um framework com abordagem híbrida que combina o modelo generativo Gemini AI e o extrator de relações REBEL com Graph Convolutional Networks (GCNs) para otimizar a classificação textual. A metodologia utiliza o Gemini para refinamento textual e extração de conhecimento estruturado em triplas, comparando sua eficácia com o modelo REBEL na construção de grafos heterogêneos. Os experimentos realizados nos datasets BBC News, Reuters-21578, AG News e 20 Newsgroups demonstraram a robustez da arquitetura proposta, com destaque para o desempenho no dataset 20 Newsgroups, que alcançou 96,91% de acurácia de validação, superando baselines como TextGCN e ChatGraph. A análise comparativa revelou que, enquanto o REBEL ofereceu maior estabilidade e eficiência na extração de triplas (87,65% de acurácia no AG News), a combinação de embeddings densos (BERT) e arestas semânticas provou-se determinante para a performance global, mitigando a necessidade de pré-refinamento textual em cenários de alta complexidade como o Reuters.

Palavras-chave: modelos de linguagem de grande escala (LLMs), classificação de texto, Gemini, redes neurais em grafos, grafos de conhecimento.

Abstract

The structured representation of human knowledge and the advancement of Large Language Models (LLMs) have driven new frontiers in Artificial Intelligence. In this context, Knowledge Graphs (KGs) emerge as essential tools for organizing complex information, enabling their integration with deep learning techniques. This work introduces GenGraph, a hybrid framework that combines the generative model Gemini AI and the relation extractor REBEL with Graph Convolutional Networks (GCNs) to optimize text classification. The methodology utilizes Gemini for text refinement and structured knowledge extraction into triplets, comparing its effectiveness with the REBEL model in the construction of heterogeneous graphs. Experiments conducted on the BBC News, Reuters-21578, AG News, and 20 Newsgroups datasets demonstrated the robustness of the proposed architecture, highlighting the performance on the 20 Newsgroups dataset, which achieved 96.91% validation accuracy, surpassing baselines such as TextGCN and ChatGraph. Comparative analysis revealed that while REBEL offered greater stability and efficiency in triplet extraction (87.65% accuracy on AG News), the combination of dense embeddings (BERT) and semantic edges proved decisive for global performance, mitigating the need for text pre-refinement in high-complexity scenarios such as Reuters.

Keywords: large language models (LLMs), text classification, Gemini, graph neural networks, knowledge graphs.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A busca por maneiras de representar e estruturar o conhecimento humano precede o surgimento das ferramentas de IA (Inteligência Artificial) modernas. As ferramentas atuais de processamento de linguagem natural (PLN) são resultado de uma extensa trajetória de pesquisa da área. O PLN é uma subárea da Inteligência Artificial voltada especificamente ao desenvolvimento de metodologias que visam analisar, compreender e extrair significado da linguagem humana de forma efetiva, permitindo que diversas tarefas e aplicações sejam executadas, tais como tarefas de classificação, que envolvem a atribuição de categorias e rótulos a documentos ou pedaços de texto com base no seu conteúdo, e refinamento de texto. Não limitante a isso, também encontra-se na literatura PLN voltada à análise de sentimento[1], aprimoramento de chatbots, aplicações assistivas, dentre outros. (AGARWAL, 2019)

Dentro do campo da PLN destacam-se os Large Language Models (LLMs), uma categoria de modelos de linguagem baseados em conjuntos de dados de larga escala treinados para compreender e processar tarefas de linguagem natural (NAVEED et al., 2025).

Uma característica vantajosa dos LLMs reside em sua capacidade de se integrar a várias outras abordagens e metodologias no campo da inteligência artificial. Um excelente exemplo pode ser visto em aplicativos generativos de IA, ferramentas de processamento de texto elaboradas com base em LLMs e modelos tradicionais de aprendizado de máquina pré-treinados, conforme elucidado por NAVEED et al.. Dentre os modelos generativos disponíveis mais consolidados no mercado destacam-se o GPT-3 e GPT-4, ambos desenvolvidos pela OpenAI[2], que possuem capacidade de gerar textos naturais e coesos, executar tarefas de tradução e revisão de texto. Além disso, no domínio dos modelos desenvolvidos pela OpenAI, temos o DALL-E 2 e DALL-E 3, ferramentas de geração de imagens com base em descrições textuais. Menções significativas também incluem modelos como o LaMDA e Gemini, desenvolvidos pela Google[3], com capacidade de gerar diálogos, criar imagens e conduzir análises de dados (ANIL et al., 2023).

Ainda no âmbito da inteligência artificial, a interseção de grafos de conhecimento e técnicas de aprendizado profundo (deep learning) emerge como uma área de pesquisa de grande impacto. Grafos de conhecimento são estruturas utilizadas para representar a relação entre diferentes tipos de entidades, como objetos e pessoas (YE et al., 2022). Dada a sua capacidade de representação multimodal de forma estruturada, os grafos de conhecimento se tornaram uma área de pesquisa essencial no contexto do subcampo do aprendizado de máquina profundo.

O aprendizado profundo, por sua vez, caracteriza-se essencialmente por redes neurais[4] de

[1] Atribuindo valores ao texto (positivo, negativo ou neutro), a análise de sentimento ajuda a identificar o teor de uma frase ou sequência de palavras

[2] https://openai.com/about/

[3] https://ai.google/build

[4] Modelo representativo do campo da computação, na qual a estrutura se assemelha à estrutura natural da rede de

multicamadas, que permitem decifrar padrões e relacionamentos complexos, além de demonstrar capacidade de lidar com problemas complexos em aplicações do mundo real, como reconhecimento de imagem, processamento de linguagem natural e diagnósticos médicos (DONG; WANG; ABBAS, 2021).

O estudo propõe um *framework* que combina o Gemini API para refinamento textual e extração de conhecimento com *Graph Convolutional Network* para executar tarefas de classificação textual. Portanto, através da aplicação dos principais conceitos e aplicações de LLMS, grafos de conhecimento e técnicas de aprendizagem profunda levantados durante a revisão bibliográfica, o trabalho propõe uma estrutura metodológica que combina a capacidade de extração de grafos de conhecimento dos modelos generativos, com a capacidade de classificação de textos das GNNs, as quais serão treinadas para classificar textos com base nos grafos de conhecimento extraídos. O desempenho das GNNs será é avaliado em diferentes bases de dados e configurações, buscando identificar a melhor combinação de hiperparâmetros e técnicas de treinamento. Essa estrutura objetivou aprimorar o desempenho da classificação em bases de dados de acesso aberto, explorando a interação entre o manejo de grandes volumes de dados textuais e o processamento de modelos baseados em grafos.

Trabalhos como os de Y. Shi et al. (2023) e Wang, Yizhao et al. (2023) destacam como a integração de LLMs e GNNs pode aprimorar a precisão e a eficácia da classificação de texto. O primeiro propõe um modelo que utiliza o ChatGPT para refinar texto, extrair conhecimento e converter esse conhecimento em grafos, que são então usados em uma rede neural de grafo (GNN) para melhorar a classificação de texto. O segundo estudo introduz o Text-FCG, que utiliza GNNs com unidades GRU para processar informações de grafos, construindo um grafo único para cada documento de texto e capturando suas relações semânticas. Ambos os trabalhos contribuem significativamente ao demonstrar como a integração de LLMs e GNNs pode aprimorar a precisão e a eficácia da classificação de texto.

Espera-se que esta pesquisa alcance relevância ao explorar o potencial da combinação do modelo generativo Gemini com Redes Neurais Baseados em Grafos (GNNs), para aprimorar tarefas de classificação textual. O framework proposto busca preencher uma lacuna no atual estado da arte da integração de GNNs-LLM, visto que, até o momento, não existem pesquisas que explorem especificamente o desempenho do Gemini em conjunto com GNNs para tarefas de classificação. A escolha do Gemini é motivada também por suas características técnicas, sendo um modelo multimodal avançado com janela de contexto ampliada e processamento eficiente de texto e imagens.

Nesse sentido, a validação experimental do trabalho demandou não apenas a implementação da arquitetura neural, mas também a construção de uma base de dados de grafos que permitisse a comparação direta entre diferentes métodos de extração. Dada a escassez de *benchmarks* prévios que consolidem extrações via Gemini/REBEL para tarefas de classificação

neurônios do cérebro humano.

supervisionada, este trabalho expande sua contribuição para além do modelo arquitetural, oferecendo um recurso prático comunidade científica ao disponibilizar um dataset inédito e público. Este recurso contém os grafos de conhecimento extraídos pelas abordagens REBEL e Gemini para os benchmarks Reuters-21578, BBC News e AG News, visando facilitar a reprodutibilidade dos experimentos e servir como base padronizada para a avaliação de futuras arquiteturas baseadas em grafos.

2 Trabalhos Relacionados

A combinação de redes neurais baseadas em grafos (GNN) com LLMs e outras técnicas de aprendizado de máquina tem sido extensamente explorada na literatura. Nesta seção apresentamos algumas abordagens notáveis que combinam GNNs, LLMs e ML, relacionadas ao framework proposto, as quais serão discutidas em seguida. Um modelo classificatório que utiliza o conhecimento extraído pelo ChatGPT para construir estruturas de grafos que permitem potencializar tarefas de classificação de texto, foi proposto por (SHI et al., 2023) no trabalho *"ChatGraph: Interpretable Text Classification by Converting ChatGPT Knowledge to Graphs"*. Neste trabalho, o *ChatGPT* é empregado para realizar refinamento de texto, extração de conhecimento e conversão do conhecimento em uma estrutura de grafo. Essa estrutura é então aplicada em uma rede neural de grafo (GNN) de uma camada para a classificação de texto. Os experimentos realizados pelos autores foram feitos utilizando alguns *datasets* comumente utilizados em tarefas de classificação de texto, os resultados evidenciaram que o método proposto supera o desempenho do *ChatGPT* em testes zero/few-shot.

A exploração das estruturas de grafos de conhecimento com tarefas de classificação também foi proposta com o framework *OPHELIA (knOwledge graPH-augmented tExt cLassIfication ApproacH)* (COSTA, 2023), uma abordagem baseadas em Redes Neurais Profundas que combina *embeddings* de palavras e de conhecimento, com o objetivo de aprimorar a presentação semântica de documentos. A metodologia realiza o treinamento do conjunto de *embeddings* utilizando a ferramenta *fastText*, e emprega técnicas como *entity recognition and linking* para anotar semanticamente a frequência de entidades nomeadas nos documentos, substituindo essas menções em representações por grafos de conhecimento. Assim, tanto as palavras quanto as entidades tem representações vetoriais e são utilizadas como entrada para os modelos de classificação.

Os resultados do framework proposto por (COSTA, 2023), durante a etapa de avaliação, mostraram-se competitivos em comparação com soluções tradicionais já consolidadas na literatura. Utilizando métricas como acurácia e F1-score, o modelo adotou uma rede neural simples com embeddings de apenas 50 dimensões, demonstrando desempenho robusto na maioria dos conjuntos de dados de domínio aberto, alcançando mais de 90% de acurácia em bases como BBC News e AG News. Esses resultados evidenciam uma lacuna nas abordagens convencionais de classificação de texto, que muitas vezes deixam de explorar recursos semânticos mais ricos, como grafos de conhecimento e embeddings conjuntos de palavras e entidades.

O artigo *"Graph embedding approaches for social media sentiment analysis with model explanation"* (S.; KRISHNA; GOVINDARAJAN, 2024), também propõe uma abordagem interessante utilizando GNN para representar dados obtidos de posts em redes sociais em forma de grafos e utilizá-los para fazer análise de sentimento. A metodologia deste trabalho se baseia em aplicar *feature extraction* utilizando três tipo de GNNs: GCN, GAT e Node2Vec. A

partir das características extraídas dos grafos, um modelo de classificação é treinado utilizando métodos como *Random Forest, Decision Tree, Support Vector Machine* e *Logistic Regression*. Os resultados dos experimentos conduzidos pelos autores utilizando GCN para representar dados em grafos superaram os métodos GAT e *Node2Vec* na análise de sentimento em redes sociais. Isso se deve à capacidade do GCN de capturar de forma mais eficiente as relações complexas entre os elementos do texto.

Outra abordagem que buscou potencializar a performance das técnicas de aprendizado utilizando grafos foi proposta no estudo *"Text FCG: Fusing Contextual Information via Graph Learning for text classification"* por (WANG et al., 2023). O trabalho propõe um método para classificação de texto que utiliza GNNs com unidades recorrentes GRU para processar a informação do grafo, chamado Text-FCG, o modelo constroi um único grafo para cada documento de texto, capturando suas relações semânticas. Os resultados dos experimentos mostram que o modelo alcançou desempenho positivo ou superior a métodos baseados em grafos como TextGCN e modelos pré-treinados como BERT. O trabalho também explora a combinação de Text-FCG com BERT (Text-FCG + BERT), demonstrando melhorias adicionais na precisão da classificação. No geral, Text-FCG se apresenta como um modelo satisfatório para classificação de texto que combina efetivamente informações contextuais por meio do aprendizado em grafo.

O TextGCN, proposto por Yao, Mao e Luo (2019), consolidou-se como o estado da arte ao modelar um corpus inteiro como um grafo heterogêneo composto por nós de palavras e documentos. Esta abordagem permitiu que a coocorrência global de palavras fosse capturada diretamente pela convolução de grafos. A relevância do TextGCN para este trabalho reside justamente em sua capacidade de estabelecer relações em nível de corpus de forma inovadora. O framework GenGraph proposto nesta dissertação constrói sobre essa base, mas busca superar suas limitações: enquanto o TextGCN se restringe à adjacência estatística baseada em frequência de palavras, a nossa arquitetura enriquece a topologia do grafo inserindo conexões semânticas explícitas extraídas via triplas de conhecimento de Modelos de Linguagem (LLMs).

3 Fundamentação teórica

A Seção 3.1 apresenta uma revisão histórica do desenvolvimento dos primeiros modelos de linguagem, juntamente com uma discussão sobre as arquiteturas introduzidas ao longo dos anos. Em seguida, na subseção 3.1.1 é apresentado alguns métodos e componentes e etapas essenciais dos Large Language Models (LLMs). Também serão detalhados os componentes e processos comuns na construção e treinamento de LLMs, como pré-processamento de dados, tokenização, pré-treinamento, ajuste de hiperparâmetros e alinhamento. A partir da seção 3.2, a revisão bibliográfica parte para a exploração dos grafos de conhecimento e suas aplicações no campo do aprendizado de máquina. A seção 3.3 introduz o aprendizado de representação de conhecimento, que engloba abordagens tradicionais e modernas de aprendizado baseado em grafos, incluindo redes neurais de grafos. Por fim, a seção 3.4 discorre sobre as técnicas e métodos de classificação de texto, bem como os processos para criar um modelo classificador.

3.1 Large Language Models

Esta seção apresenta uma breve introdução a respeito dos principais modelos de linguagem, bem como uma discussão a respeito das arquiteturas introduzidas ao longo dos anos. Também é descrito alguns componentes e etapas essenciais dos Large Language Models.

Os avanços na modelagem de linguagem natural evoluíram significativamente, desde os modelos estatísticos e n-gram até os atuais LLMs (*Large Language Models*), impulsionados pela introdução dos *Transformers*. Essa arquitetura revolucionou o campo ao permitir processamento paralelo e capturar relações contextuais em grandes volumes de dados. Nos últimos anos, empresas como Google e OpenAI lançaram modelos de destaque, como T5, GLAM, LaMDA, GPT-3, GPT-4 e Bard, cada um aprimorando a capacidade de compreensão e geração de texto. Em 2023, o Google apresentou o Gemini, um modelo multimodal avançado com janela de contexto ampliada e processamento eficiente de texto e imagens, consolidando-se como uma ferramenta escalável e versátil para aplicações diversas.

Para compreender o funcionamento dos LLMs e seu avanço nos últimos anos, é essencial analisar os componentes e processos que possibilitam seu desenvolvimento. Apesar da diversidade de modelos disponíveis no mercado, muitos compartilham uma estrutura comum, baseada em técnicas fundamentais de processamento de linguagem natural. Dessa forma, a próxima seção explora em detalhes os principais elementos envolvidos na construção desses modelos, destacando desde a preparação dos dados até o ajuste fino e o alinhamento, que garantem sua eficácia e confiabilidade.

3.1.1 Passos para Aplicação dos LLMs

Embora exista uma quantidade significativa de modelos de linguagem no mercado, treinados para diferentes tarefas ou incorporados em aplicativos generativos, a maioria uti-

liza componentes similares. Nesta seção, é feita uma descrição detalhada dos componentes e processos popularmente utilizados nos LLMs.

Alguns elementos e procedimentos fundamentais envolvidos na implementação de um Modelo de Linguagem (LLM) abrangem pré-processamento e limpeza de dados, tokenização, pré-treinamento do modelo, ajuste de hiperparâmetros e alinhamento (MINAEE et al., 2024).

A. Pré-processamento e limpeza de dados:

O pré-processamento e a limpeza de dados são etapas significativas no desenvolvimento de um modelo de alto desempenho, garantindo a consistência e a integridade das informações que serão utilizadas. As técnicas focais para o refinamento de texto incluem:

- **Remoção de ruído:** eliminação de dados irrelevantes ou "ruídos"que possam impactar negativamente a performance do modelo.
- **Pré-processamento de texto:** padronização e limpeza de dados textuais, com a remoção de erros de sintaxe, pontuação, grafia e outros elementos irrelevantes para o aprendizado do modelo.

B. Tokenização

A tokenização é o processo que divide uma sequência textual em unidades individuais, denominadas tokens (JIN, 2022). Essas unidades fragmentadas podem ser palavras inteiras, partes de palavras, caracteres ou qualquer outro elemento relevante no contexto da análise, sendo um passo fundamental para o processamento da linguagem pelo modelo.

C. Pré-treinamento

O pré-treinamento é uma etapa essencial no desenvolvimento de Modelos de Linguagem de Grande Escala (LLMs), na qual o modelo processa grandes quantidades de dados não rotulados de forma semi-supervisionada para aprender a estrutura e os padrões subjacentes da linguagem, melhorando sua capacidade de generalização e desempenho em tarefas futuras (MINAEE et al., 2024). As abordagens mais comuns para esta etapa são a modelagem de linguagem autorregressiva (autoregressive language modeling), que treina o modelo para prever o próximo token com base nos anteriores, e a modelagem de linguagem mascarada (masked language modeling), que o treina para prever tokens que foram omitidos ("mascarados") com base em seu contexto circundante. Ambas as técnicas são fundamentais para estabelecer uma base sólida de conhecimento nos LLMs, permitindo-lhes capturar dependências contextuais complexas.

D. Ajuste e Otimização de Modelos

O ajuste de modelos é um processo que visa aprimorar o desempenho em tarefas específicas. No contexto deste trabalho, a otimização não envolve o re-treinamento (fine-tuning) do

LLM, que possui um alto custo computacional, mas sim a aplicação de duas estratégias distintas para os diferentes componentes da arquitetura:

No caso do LLM (Gemini), a abordagem utilizada é o ajuste de instrução (instruction tuning). O foco dessa técnica está na formulação de um prompt que especifica claramente a tarefa que o modelo deve executar. Isso permite direcionar o LLM para gerar as saídas desejadas com base nas instruções fornecidas, aproveitando seu conhecimento pré-treinado sem a necessidade de modificar seus parâmetros internos.

Já o ajuste de hiperparâmetros é a estratégia aplicada especificamente ao modelo GCN, conforme detalhado posteriormente na metodologia. Este processo consiste na otimização de parâmetros da arquitetura da rede para refinar sua performance na tarefa específica para a qual foi designada.

E. Alinhamento

A capacidade de execução de tarefas dos modelos de linguagem atuais vai muito além da correção ortográfica ou resumo, abrangendo também a geração de textos extensos e complexos. Nesse sentido, a etapa de alinhamento dos LLMs diz respeito ao processo de garantir que as informações geradas pelos modelos estejam alinhadas com os valores éticos humanos (MINAEE et al., 2024). A urgência do alinhamento se intensifica com o potencial dos LLMs de gerar conteúdo enviesado, inconsistente e até mesmo irreal, tornando essencial assegurar que suas ferramentas operem em conformidade com o bem-estar da humanidade.

Com os avanços na área e o surgimento dos LLMs, a discussão sobre alinhamento tornou-se central. À medida que esses modelos crescem em escalabilidade e complexidade, aumenta a necessidade de que mantenham transparência. Embora a definição do que é moral e ético seja complexa, o objetivo central desse processo é garantir que as ferramentas de IA sirvam à humanidade de forma positiva e responsável.

3.2 Grafos de Conhecimento

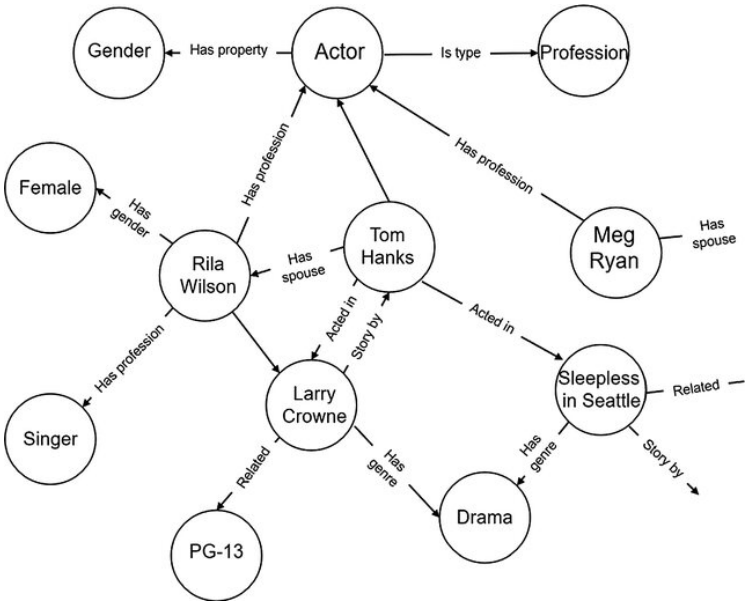
Historicamente, o primeiro registro de representação do conhecimento data de 1959, com o conceito de rede semântica, proposto por Richard Richens, pesquisador da Universidade de *Cambridge* (JI et al., 2021). Essa proposta visava representar as relações entre as palavras da linguagem, reconhecendo sua natureza linear (sequências de palavras) e a possibilidade de representação por meio de ramificações (CORONA; HERNáNDEZ-COLíN; BUSTILLO-HERNáNDEZ, 2012).

Inicialmente, o conceito de representação do conhecimento seguia uma abordagem linear baseada em símbolos. Mais tarde, esse método foi explorado no campo da inteligência artificial com o objetivo de alcançar a interoperabilidade semântica, ou seja, adaptar e treinar as máquinas para que possam compreender e utilizar a linguagem de acordo com a compreensão humana. Essa

interoperabilidade pôde ser alcançada por meio de ontologias, que são essencialmente modelos formais de representação do conhecimento. Esses modelos possuem estruturas semelhantes às redes semânticas, em que conceitos ou entidades são representados como nós, e as relações entre eles, por arcos rotulados que definem a natureza da conexão (CORONA; HERNáNDEZ-COLíN; BUSTILLO-HERNáNDEZ, 2012).

Com a evolução dessas abordagens, o desenvolvimento de ferramentas avançadas, como o *Google Knowledge Graph*, tornou-se um marco significativo. Em 2012, a introdução do *Google Knowledge Graph* impulsionou o crescimento exponencial dessa área, tanto no meio acadêmico quanto na indústria. Essa ferramenta apresentou-se como uma estrutura baseada em grafos de conhecimento de grande escala, hospedada pelo *Google*, com o objetivo de otimizar o mecanismo de pesquisa do *Google Search*. A pesquisa, que antes era baseada em strings[1], passou a ser baseada em entidades (HOGAN et al., 2021), conforme ilustrado na Figura 1. Por exemplo, em vez de simplesmente corresponder à palavra "Louvre" durante a pesquisa, o novo mecanismo compreenderia que a consulta refere-se a um conceito/entidade: o *Museu do Louvre*, um museu de arte localizado em Paris, França.

Figura 1 – Representação visual do graph de conhecimento original publicado pela Google em 2012



Fonte: (ZHU et al., 2020)

Ainda no âmbito da inteligência artificial, os grafos de conhecimento (KGs) se consolidaram como ferramentas poderosas para organizar e representar informações complexas de forma estruturada e interligada. A literatura apresenta diversas definições e conceitos para "Grafo de Conhecimento" que visam sintetizar suas características principais. Seguindo a definição de (WANG et al., 2017), um grafo de conhecimento define-se como um modelo multimodal composto por entidades e relacionamentos. As entidades podem corresponder a objetos ou con-

[1] Sequência de caracteres, geralmente utilizada para representar palavras, frases ou textos em um programa

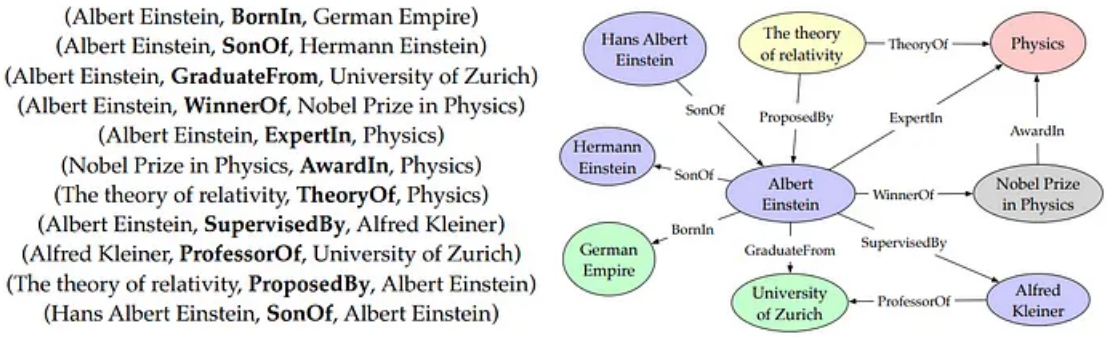
ceitos do mundo real, enquanto os relacionamentos, representados por arestas, conectam essas entidades e expressam as interações entre elas. Essa estrutura multimodal permite a representação de informações de diversas naturezas, incluindo texto, imagens, áudio e vídeo.

Utilizando notação matemática, resume-se que $G = E, R, F$, onde E, R e F equivalem, nessa ordem, entidades, relacionamentos e fatos. Fatos são comumente representados por triplas, compostas por (*subject, predicate, object*), em que o *subject* e *object* são as entidades e o *predicate* é a associação entre elas. Por exemplo, na tripla (“Machado de Assis”, “nasceu em”, “Rio de Janeiro”), “Machado de Assis” é o sujeito, “nasceu em” é o predicado, e “Rio de Janeiro” é o objeto.

As triplas são fundamentais na representação do conhecimento, pois permitem o entendimento intuitivo das relações complexas entre as entidades. Além disso, sua natureza de conectar conceitos e suas semânticas facilita a integração de dados e recuperação de informações em aplicações de diversos tipos (GAO et al., 2018). Isso é especialmente útil em aplicações de motores de busca ou sistemas de recomendação que necessitam inferir relações implícitas para melhorar a precisão e eficiência dos resultados (WANG et al., 2017).

Nesse sentido, o *knowledge graph* (KG) e *knowledge base* (KB) se assemelham ao armazenar correlações semânticas, mas se diferem em sua estrutura. O KG organiza essas informações por meio de representações grafais, enquanto a KB se concentra no armazenamento de interpretações e inferências sobre os fatos. A Figura 2 evidencia essa diferença, na esquerda temos a representação das entidades e relações entre eles através de uma tripla e na direita temos a mesma representação das conexões entre as entidades em forma de grafo.

Figura 2 – Representação visual de como um knowledge base (à esquerda) pode ser transformado em um knowledge graph (à direita)

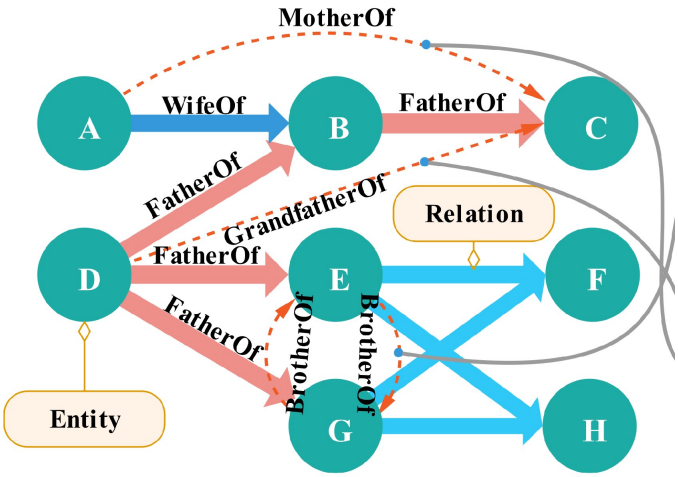


Fonte: (ZHOU et al., 2020)

Ampliando-se o conceito descrito anteriormente, a Figura 3 ilustra outro exemplo de conexões semânticas entre as entidades. Nela, podemos observar os nós, que representam indivíduos e as arestas representam relações entre eles. Os rótulos das arestas indicam o tipo de relação, como "Mãe de", "Esposa de", "Pai de", "Irmão de"e "Avô de". Logo, de acordo com a

notação descrita anteriormente temos que, $V = \{A, B, C, D, E, F\}$, é o conjunto de nós representando indivíduos (A, B, C, D, E, F) e $E = \{(A, C), (A, B), (B, F), (C, D), (C, E), (E, D)\}$ é o conjunto de arestas representando relações entre indivíduos. Utilizando a representação por triplas, temos que, $\{A, wife/esposa, B\}$.

Figura 3 – Representação visual de uma estrutura complexa de conhecimento em grafo



Fonte: (SUN et al., 2022)

Trabalhos como (CABOT; NAVIGLI, 2021), introduzem a extração otimizada de grafos de conhecimento a partir de texto utilizando o modelo BART. O REBEL *(Relation Extraction By End-to-end Language generation)*, propõe uma abordagem para extrair representações de triplas de grafos de conhecimento a partir de resumos extraidos do Wikipedia, conectando as entidades encontradas nos textos processados com aos respectivos correspondentes do *Wikipedia*, o que possibilita a extração das relações entre essas entidades. Ao enquadrar a Extração de Relações (RE) como uma tarefa seq2seq (sequência a sequência), o REBEL decodifica eficientemente as relações utilizando o modelo BART, uma arquitetura baseada em transformadores, conhecida por suas capacidades bidirecionais e autoregressivas. O modelo BART desempenha um papel crucial na garantia da geração eficaz de tripletas de relações a partir de sentenças de entrada, tendo demonstrado desempenho de ponta em diversos benchmarks. Além disso, o REBEL mantém sua adaptabilidade ao integrar-se com qualquer dump da Wikipédia em múltiplas línguas, permitindo que o sistema se mantenha atualizado com as alterações na Wikipédia e no Wikidata, garantindo que os conjuntos de dados extraídos permaneçam atuais e relevantes.

A estrutura multimodal e relacional dos grafos de conhecimento (KGs) oferece diversas vantagens em comparação com outras formas de representação de conhecimento, como bases de dados tradicionais. A capacidade de interligar informações de diversas naturezas e expressar relações complexas entre entidades permite que os KGs modelem o mundo real de forma mais rica e precisa, facilitando a análise e o processamento de informações (WU et al., 2022). Essa representação completa e interligada permite uma visão contextualizada dos dados,

potencializando a inferência de informações.

Dentre das vertentes do estudo do grafo de conhecimento, temos: *Knowledge Representation Learning (KRL), Knowledge Aware Applications (KAW), Knowledge Acquisition (KA), Temporal Knowledge Graph (TKG)* (JI et al., 2022). O KRL processa as associações entre as entidades a fim de inferir conhecimento em representações semânticas distribuídas em vetores de baixa dimensão. Essas representações contribuem para a análise e processamento do conhecimento armazenado no grafo, facilitando o entendimento de todas as conexões e interconexões entre os dados. Atualmente na literatura, encontra-se KRL em aplicações de diversas áreas como processamento de imagem, reconhecimento de fala, processamento de linguagem natural e aplicações de redes de informação (WU et al., 2022).

O *KAW (Knowledge Aware Applications)* agrupa técnicas que utilizam uma metodologia para gerenciar o conhecimento de forma efetiva, buscando criar fluxos de trabalhos para resolver problemas do mundo real. Alguns exemplos de aplicações são entendimento de linguagem natural[2], sistemas de recomendação e *question answering (QA)* (JI et al., 2022).

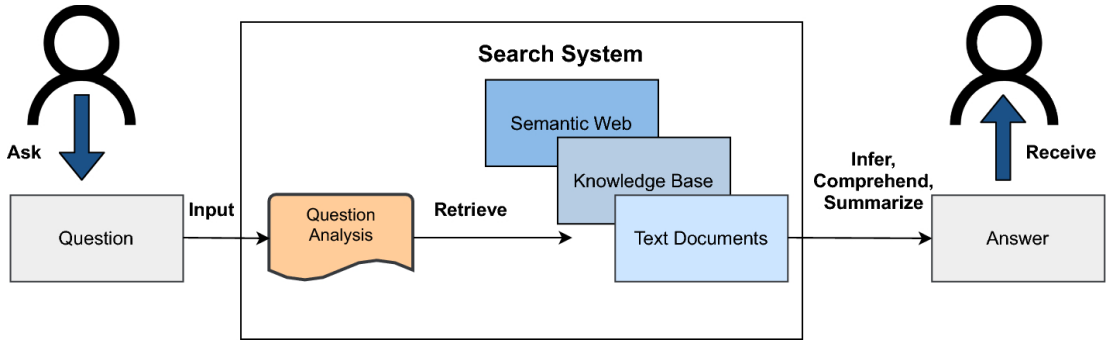
No que tange o entendimento de linguagem natural, por exemplo, as técnicas de KAW aprimoram a estrutura do conhecimento ao extrair informações significativas de dados não estruturados. Por meio da análise aprofundada de textos não estruturados, os sistemas podem identificar padrões, relações e contextos, permitindo a compreensão de comandos em linguagem natural, fornecendo saídas relevantes e precisas para os usuários.

A integração de grafos de conhecimento com sistema de recomendação, por sua vez, permite que problemas de esparsidade sejam mitigados. Esses problemas surgem quando os sistemas de recomendação não dispõem de dados suficientes para identificar e sugerir itens relevantes aos usuários. Para superar essa limitação, os grafos de conhecimento são utilizados para utilizar *common sense reasoning*, ou seja, fazer predições a partir do conhecimento adquirido. Por fim, sistemas *question-answering (QA)* baseados em grafos de conhecimento respondem a perguntas feitas em linguagem natural com fatos provenientes dos KGs. Nesse contexto, alguns métodos também realizam a injeção de conhecimento simbólico para promover o *common sense reasoning*, ampliando assim a capacidade de compreensão e resposta dos sistemas de QA.

A Figura 4 ilustra de forma esquemática como funciona a arquitetura da maioria dos sistemas de QA, onde um mecanismo de busca identifica informações de diferentes fontes. O pipeline começa com o usuário enviando uma pergunta como entrada (*Ask* $\rightarrow$ *Question* $\rightarrow$ *Input*), o sistema de busca (*Search System*) analisa e processa a pergunta de entrada (*Question Analysis*) e recupera as informações necessárias, como semântica (*Semantic Web*), conhecimento-base (*Knowledge Base*) e documentos de texto (*Text Documents*), para inferir conhecimento (*Infer, Comprehend, Summarize*) e ser capaz de responder a pergunta do usuário (*Answer* $\rightarrow$ *Receive*).

[2] um subtópico da PLN (Processamento de Linguagem Natural) que trata da compreensão de leitura por máquina.

Figura 4 – Representação visual da arquitetura comum de um sistema de QA



Fonte: (ZAIB et al., 2022)

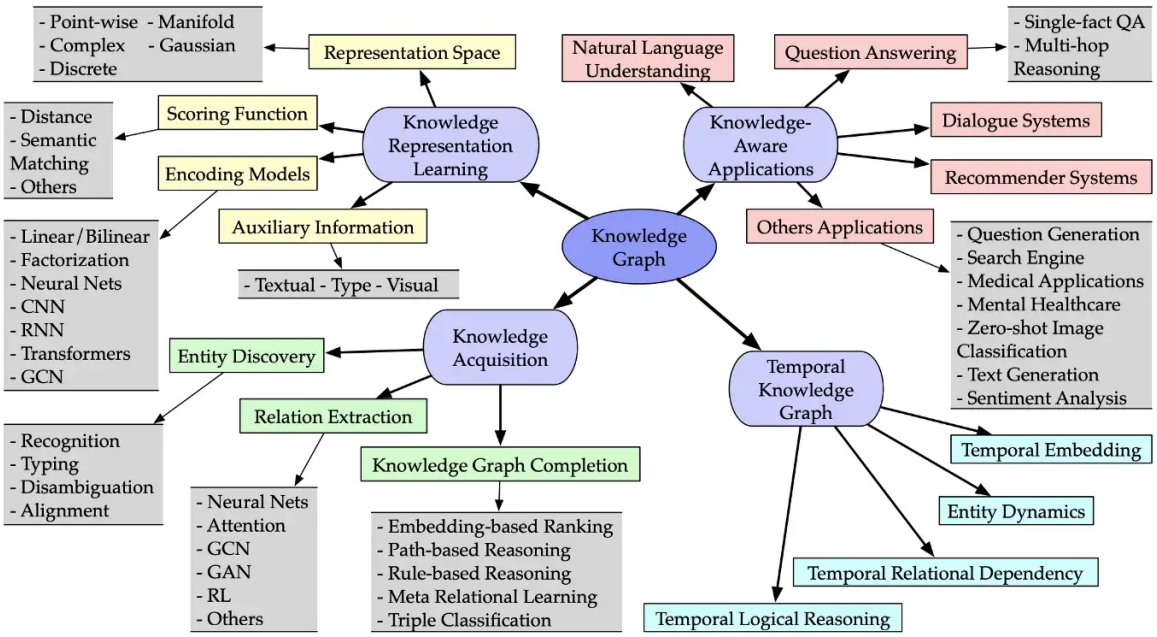
A terceira vertente no campo do estudo do grafo de conhecimento é o *Knowledge Acquisition (KA)*, e refere-se a etapa de analisar e interpretar um conhecimento ou corpus não estruturado e transformá-lo em um conhecimento estruturado, extraindo suas entidades e associações. As tarefas do KA são divididas em três categorias: *knowledge graph completion (KNC)*, extração de relações e descoberta de entidades. O primeiro diz respeito a expandir grafos de conhecimento existentes, enquanto os outros são sobre descobrir novos conhecimentos a partir de um texto (*corpus*). A etapa de extração de entidades inclui reconhecimento, desambiguação, classificação e alinhamento. O reconhecimento envolve a identificação de entidades através de menções no texto; a desambiguação visa mitigar as ambiguidades em relação a quais entidades específicas estão sendo referenciadas; a classificação categoriza e tipifica as entidades; e o alinhamento mapeia as entidades extraídas para uma base de conhecimento existente (JI et al., 2022).

Por outro lado, a extração das relações envolve processos mais complexos, dentre eles, mecanismo de atenção, redes convolucionais de grafos (GCNs), treinamento adversarial, aprendizado por reforço, aprendizado residual profundo e aprendizado de transferência (JI et al., 2022). Além dessas técnicas, outras abordagens emergentes e metodologias estão sendo exploradas no campo da KA, como o uso de modelos pré-treinados de linguagem natural (como *BERT* e *GPT*) para aprimorar a extração de entidades e relações.

Por fim, a última vertente é o *Temporal Knowledge Graph (TKG)*, o qual é considerado uma extensão do KG em uma dimensão temporal, onde cada representação do conhecimento é armazenado em um quádruplo contendo uma variável adicional que denota o tempo válido para cada fato. Essa adição temporal permite que os grafos de conhecimento evoluam de uma abordagem que lida apenas com fatos estáticos para uma que acomoda fatos dinâmicos, refletindo mudanças ao longo do tempo.

A Figura 5 exibe todas as categorias e subcategorias citadas anteriormente, bem como as técnicas mais comumentes utilizadas no estudo de KGs. No contexto deste trabalho, o foco gira em torno do *Knowledge Representation Learning (KRL)* e seus métodos, com ênfase especial nas *Graph Neural Networks (GNNs)*.

Figura 5 – Representação visual das categorias e subcategorias do estudo em Knowledge Graphs (KGs)



Fonte: (ZHOU et al., 2020)

3.3 Knowledge Representation Learning

Com o desenvolvimento de novas pesquisas no âmbito de Knowledge Graphs (KGs) e o avanço do aprendizado profundo, o KRL (Knowledge Representation Learning) se apresentou como uma alternativa para viabilizar a utilização de KGs em aplicações do mundo real, superando os desafios de custo computacional e dispersão de dados. O KRL se concentra principalmente no processo de incorporação vetorial dos grafos de conhecimento. Esse processo envolve a transformação de entidades e relacionamentos em vetores — representações numéricas que podem ser manipuladas por algoritmos de aprendizado de máquina — com o objetivo principal de preservar o significado semântico durante a transformação. Ao criar uma representação de dados mais densa e compacta, o KRL consegue reduzir a complexidade computacional, facilitando a análise eficiente dos grafos de conhecimento (LIN et al., 2018b).

Dentre as diversas abordagens para realizar essa incorporação, as Graph Neural Networks (GNNs), se destacam como uma solução projetada especificamente para trabalhar com grafos, atuam como uma espécie de decodificador de grafos, onde entidades e relacionamentos são analisados a fim de fazer previsões com base em suas associações, preservando a estrutura geral do grafo (WU et al., 2022). As redes neurais baseadas em grafos vêm se expandindo na ciência, sendo aplicadas em sistemas de recomendação, computação visual, processamento de linguagem natural, cibersegurança e em muitos outros campos da inteligência artificial e aprendizado profundo.

Ainda no que diz respeito ao estudo do KRL (*Knowledge Representation Learning*),

podemos destacar três principais classificações das abordagens de aprendizado de representação de grafos, são estas: *traditional graph embedding* (incorporação de grafo tradicional), *modern graph embedding* (incorporação de grafo moderno) e *graph neural networks* (rede neural baseada em grafo) (WU et al., 2022).

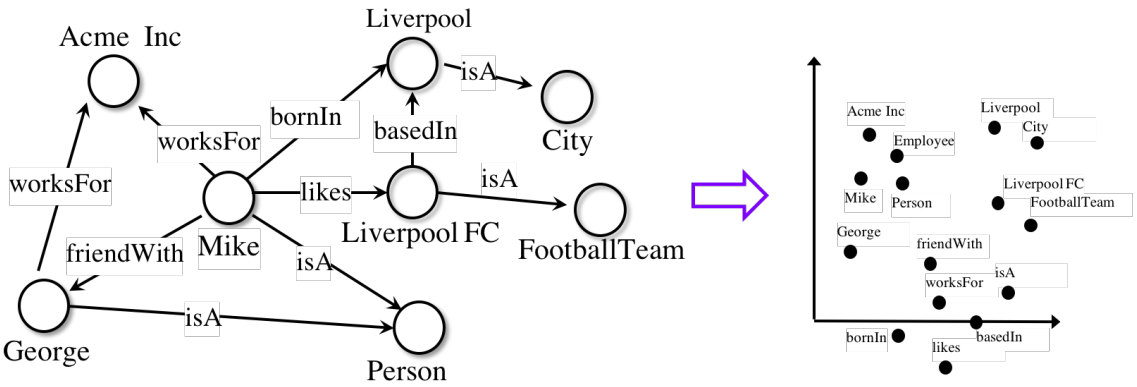
3.3.1 Traditional e Modern Graph Embedding

A incorporação de grafos tradicionais visa reconstruir a estrutura original de um grafo em um espaço vetorial de menor dimensão, preservando suas propriedades estruturais essenciais através de técnicas de inferência. As abordagens tradicionais, como Laplacian Eigenmaps e Locally Linear Embedding (LLE), focavam em grafos onde a proximidade entre os nós era bem definida, utilizando principalmente decomposição de matriz para a redução de dimensionalidade (WU et al., 2022).

No entanto, a pesquisa recente foca em grafos modernos, que são mais complexos e ricos em informações, como redes sociais ou biológicas, onde as relações entre nós não são explicitamente definidas por pesos. O objetivo do aprendizado de representação nesses grafos é capturar um conjunto mais rico de informações para permitir uma inferência mais robusta. Para isso, os métodos modernos buscam preservar diferentes aspectos da estrutura do grafo (Figura 6), como:

- **Estruturas de vizinhança e de alta ordem:** Preservar tanto as conexões diretas entre um nó e seus vizinhos imediatos quanto as relações indiretas e as estruturas de comunidades (grupos de nós densamente conectados).

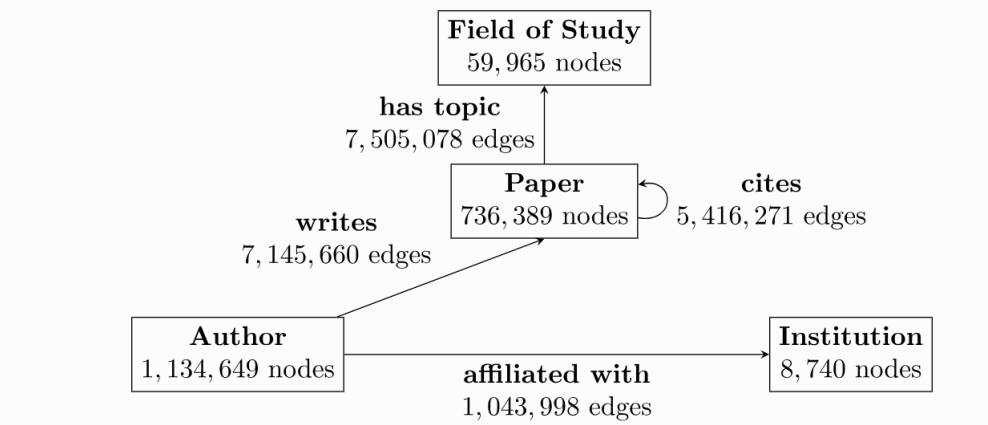
Figura 6 – Representação do processo de transformação dos nós e arestas de um grafo de conhecimento em representações vetoriais em um espaço métrico



Fonte: (COSTABELLO et al., 2019)

- **Informação lateral (side information):** Incorporar o conteúdo associado aos nós, como atributos, rótulos ou descrições textuais. Isso é especialmente relevante em grafos heterogêneos, que possuem diferentes tipos de nós e arestas, como exemplificado na Figura 7.

Figura 7 – Representação visual de um grafo heterogêneo



Fonte: (HU et al., 2021)

- **Informação de tarefas específicas:** Utilizar dados supervisionados ou semi-supervisionados para conectar as representações vetoriais dos nós a uma tarefa-alvo, permitindo o desenvolvimento de soluções mais eficientes e especializadas.

A seção a seguir aborda as Redes Neurais de Grafos (GNNs), destacando como essa tecnologia lida com a complexidade de dados estruturados em grafos para facilitar a descoberta de padrões, predições e análises interpretativas. São discutidos os desafios computacionais e a necessidade de novas técnicas de aprendizado de máquina para processar eficientemente esses dados. Além disso, a seção explora as metodologias avançadas das GNNs, como as *Graph Convolutional Networks* (GCNs), *Graph Attention Networks* (GATs) e *Graph Recurrent Networks* (GRNs), e suas aplicações no mundo real.

3.3.2 Graph Neural Networks

A estruturação de dados em grafos auxilia na compreensão de sistemas complexos, o processamento efetivo dessas informações facilita tarefas como descoberta de padrões, predição, análises interpretativas dos dados, etc. Contudo, o processo de estruturação visual de grafos apresenta alguns desafios como complexidade computacional, baixo paralelismo e dificuldade de intersecção com método de aprendizado de máquina (WU et al., 2022).

O primeiro grande desafio diz respeito à complexidade computacional envolvida no processamento iterativo de cada nó do grafo. Essa análise geralmente é feita através de uma combinação probabilística, o que demanda custo computacional que dificulta a aplicabilidade em sistemas de larga escala. A necessidade de cálculos intensivos para capturar as relações entre os nós torna o processo de análise mais lento e menos eficiente, especialmente quando se lida com grafos de grande escala.

No contexto de escalabilidade, também surgem dificuldades significativas em termos de computação distribuída. Cada nó de um grafo está intimamente acoplado aos outros, o que dificulta a distribuição dos nós em diferentes servidores. Essa dependência intrínseca aumenta

o custo de comunicação, tornando a propagação de nós através de servidores distribuídos um processo caro e ineficiente.

Ainda observando essa dependência entre os nós de um grafo, destaca-se a inviabilidade de aplicar algumas técnicas de aprendizado de máquina. Muitos métodos assumem que os dados podem ser representados em vetores independentes em um espaço. Embora a representação de grafos em vetores seja uma solução teórica, na prática, a alta dimensionalidade e a complexidade das conexões entre nós dificultam a implementação eficiente. Isso exige o desenvolvimento de novas técnicas de processamento e representação que possam lidar com a escalabilidade e a complexidade intrínseca dos grafos, integrando de forma mais eficaz os métodos de aprendizado de máquina para explorar o potencial completo dos dados estruturados em grafos (WU et al., 2022).

Para enfrentar os desafios na estruturação e processamento de grafos, as pesquisas recentes no campo das Redes Neurais de Grafos (GNNs) têm buscado mitigar essas dificuldades através do desenvolvimento de metodologias avançadas de aprendizagem de representação de grafos. Essas metodologias envolvem aprender representações vetoriais densas e contínuas de baixa dimensionalidade para os nós, de modo que o ruído ou informações redundantes possam ser reduzidos e a informação estrutural intrínseca possa ser preservada. As GNNs são projetadas para aprender representações que capturam a complexa estrutura dos grafos. Ao transformar os elementos da estrutura do grafo (nós, arestas e relações) em vetores densos de baixa dimensionalidade, as GNNs permitem uma redução significativa da complexidade e dimensionalidade dos dados. Isso facilita a aplicação de algoritmos de aprendizado de máquina, pois as representações vetoriais são mais manejáveis e eficientes para processamento.

Avanços recentes na área de aprendizado de grafos aprimoraram significativamente a capacidade das GNNs. Hoje, muitas aplicações do mundo real já utilizam GNNs para executar diversas tarefas, como detecção de *fake news*, predição de tráfego e sistemas de recomendação, entre outros (YE et al., 2022). Esses avanços têm demonstrado a eficácia das GNNs em capturar a estrutura subjacente dos dados de forma que métodos tradicionais de aprendizado de máquina não conseguem.

Conceitualmente, uma rede neural de grafo pode ser definida como uma transformação otimizada de todos os elementos da estrutura do grafo que preserva a simetria do grafo. Segundo (ZHOU et al., 2020), as GNNs são projetadas para obter os mesmos resultados independentemente da ordem dos nós no grafo. Isso significa que, ao contrário de outros métodos que podem ser sensíveis à permutação dos nós, as GNNs mantêm a integridade e a simetria das relações estruturais, assegurando que as propriedades topológicas sejam preservadas.

De modo geral, as etapas para a construção de uma GNN são estas: definição de estrutura do grafo, especificação do tipo e escala do grafo, criação da função de perda e construção do modelo (ZHOU et al., 2020).

I. Definição da estrutura do grafo

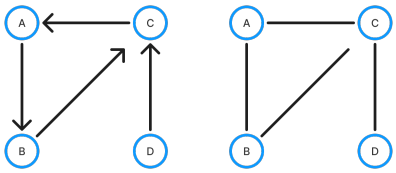
A primeira etapa diz respeito à exploração do escopo do problema que se deseja resolver utilizando GNNs, com o objetivo de identificar a estrutura apropriada do grafo. Nesta fase, pode haver dois cenários distintos: o primeiro em que a estrutura do grafo diante do escopo é explícita, como aplicações no campo da biologia (representação de moléculas), física, grafos de conhecimento, dentre outros. No segundo cenário a estrutura do grafo não é fornecida explicitamente nos dados, é necessário construir a partir dos dados brutos, de acordo com a tarefa que se deseja resolver. Isso envolve a criação de um grafo baseado nas relações inferidas dos dados, de acordo com a tarefa específica que se deseja resolver. Por exemplo, em análise de redes sociais, as conexões entre usuários podem ser determinadas a partir de interações como mensagens ou seguidores.

II. Especificação do tipo e escala do grafo

A segunda etapa do *pipeline* é a descoberta do tipo e escala do grafo. Conforme evidenciado por (ZHOU et al., 2020), grafos podem ser do tipo dirigidos/não-dirigidos, homogêneos/heterogêneos e dinâmicos/estáticos:

- Grafos dirigido/não-dirigidos: Em um grafo dirigido, as arestas direcionam o caminho que se deve tomar para percorrer os nós conectados, por outro lado, grafos não-dirigidos não mostram a direção que deve ser tomada para percorrer entre os nós, de forma que cada nó permite direcionar para qualquer aresta adjacente. A Figura 8 exemplifica a diferença entre os dois tipos, no grafo esquerdo é possível visualizar o direcionamento dos nós ($D \rightarrow C \rightarrow A \rightarrow B \rightarrow C$), enquanto o grafo direito não exibe as setas, o que significam que os nós A, B, C, D podem ser acessados por qualquer direção.

Figura 8 – Representação visual de grafos dirigidos e não dirigidos



Fonte: Elaborado pelo autor

- Grafos homogêneos/heterogêneos: Um grafo é definido como homogêneo quando possui nós e arestas do mesmo tipo, caso contrário é categorizado como heterogêneo;

- Grafos dinâmicos/estáticos: Grafos dinâmicos tem suas características e topologia alteradas de acordo com o passar do tempo;

A escala refere-se ao tamanho do grafo, que pode variar desde pequenos grafos com poucos nós até grandes grafos com milhões de nós e arestas. A escalabilidade do modelo deve ser considerada, especialmente para grafos grandes, para garantir eficiência computacional.

É importante destacar que as categorias descritas acima não são mutuamente exclusivas, ou seja, um grafo pode ter uma combinação de um ou mais tipos.

III. Definição da função de perda

A função de perda (*loss function*) é utilizada para calcular a distância entre a saída atual do algoritmo e a saída esperada, ou seja, é uma maneira de dizer quão bem o algoritmo modela o conjunto de dados. No aprendizado baseado em grafos, essa função é definida com base no tipo de tarefa e na configuração do treinamento.

Em relação ao primeiro critério, distinguem-se três diferentes tipos de tarefas:

IV. Construção do modelo

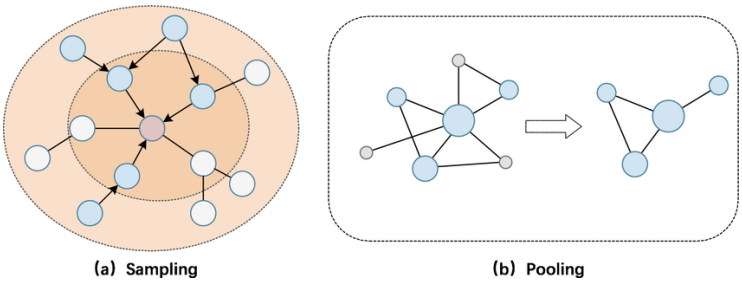
A etapa final do *pipeline* da construção de uma GNN, envolve a utilização de diversos módulos computacionais. Os módulos mais comumente utilizados são:

- Módulo de propagação: no qual tem a função de propagar as informações entre os nós no grafo para que as informações agregadas capturem tanto características quanto informações topológicas. Um dos operadores mais utilizados para esse módulo são os *convolution operators*, funções matemáticas utilizadas para processar dados aplicando um filtro sobre uma série de dados de entrada para produzir uma série de dados de saída. Nas *ConvGNNs*, são geralmente usados para extrair características de imagens, onde um filtro/*kernel* (matriz de valores) é aplicado a uma imagem de entrada, gerando uma imagem de saída onde cada pixel é uma combinação linear dos pixels na vizinhança do filtro.
- Módulo de amostragem: facilita a propagação de informações em grafos muito grandes e extensos, visto que em modelos GNN, as mensagens são agregadas em cada nó a partir do vizinho da camada anterior, o que faz com que, a cada camada, os vizinhos vão crescendo exponencialmente. Portanto, a amostragem alivia essa "explosão de vizinhos", tornando o processo gerenciável (ZHOU et al., 2020). A amostragem geralmente é combinada com o módulo de propagação para operar de maneira mais eficiente;
- Módulo de *pooling*: necessário para extrair representações de subgrafos ou grafos de alto nível, ajudando a reduzir o grafo a estruturas menores a cada iteração;

A Figura 9 exibe como funciona a manipulação dos grafos nos modulos de amostragem

e *pooling*, no grafo a esquerda é possível notar que os nós vizinhos são agregagos em camadas e no exemplo a esquerda o grafo é reduzido a uma escala menor ao inferir representações de grafos de alto nível.

Figura 9 – Representação visual de diferentes aplicações da GNN



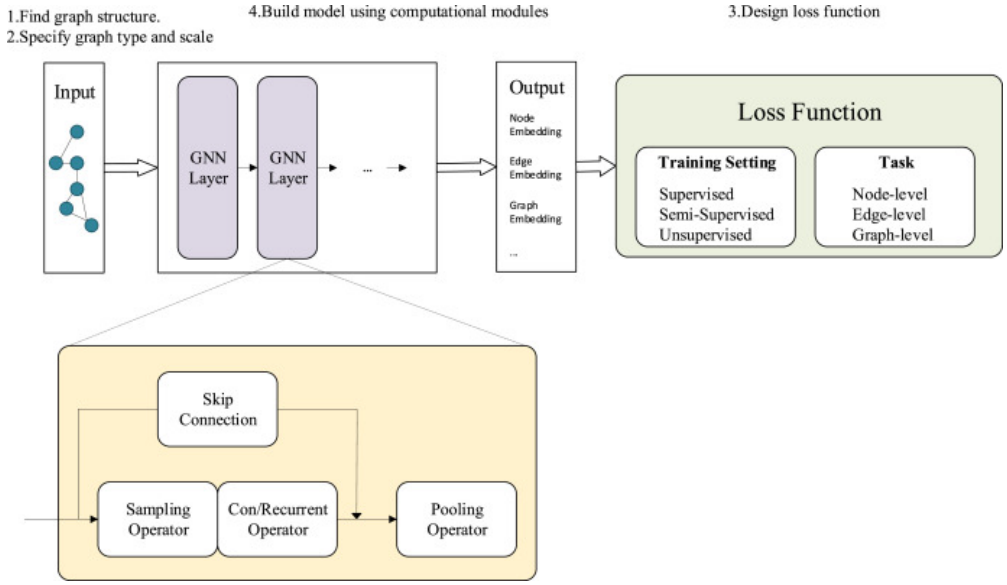
Fonte: (SUN et al., 2023)

É comum que a maioria das arquiteturas combinem um ou mais módulos descritos anteriormente em múltiplas camadas empilhadas. Essa combinação permite a obtenção de representações mais detalhadas e refinadas dos grafos, aprimorando a capacidade do modelo de capturar a profundidade das relações entre os nós e suas características (ZHOU et al., 2020).

A figura 10 exibe o *pipeline* completo das etapas de construção de uma GNN, conforme descrito nas seções anteriores. Começando pela definição da estrutura do grafo (passo 1), partindo para a especificação do tipo e escala do grafo (passo 2), e definição da função de perda (passo 3), que deve ser escolhida com base na natureza da tarefa que o modelo irá executar. A seção correspondente à *Loss Function* na figura exemplifica os diferentes paradigmas de treinamento possíveis—supervisionado, semi-supervisionado e não supervisionado—bem como os níveis de tarefa abordados pela GNN, que podem ocorrer em nível de nó, de aresta ou de grafo.

O Passo 4 encerra o processo com a construção da GNN, utilizando módulos computacionais específicos. Na etapa de saída (*Output*), são exemplificadas as representações vetoriais esperadas, que incluem *node embeddings* (representações vetoriais de nós), *edge embeddings* (representações vetoriais de arestas) e *graph embeddings* (representações vetoriais de grafos). Além disso, a figura detalha as operações internas realizadas nas camadas da GNN (*GNN Layer*), incluindo operadores recorrentes, operadores de *pooling* e conexões residuais (*skip connections*), evidenciando a complexidade computacional envolvida no processamento dos dados estruturados em grafos.

Figura 10 – Representação do processo completo com todas as etapas da construção de uma GNN



Fonte: (ZHOU et al., 2020)

Outras variações das Redes Neurais em Grafos (GNNs) têm demonstrado desempenho notável em diversas tarefas de aprendizado profundo. Especificamente, as *Graph Convolutional Networks* (GCNs), também conhecidas como *ConvGNNs*, representam um tipo de GNN inspirado nas redes neurais convolucionais tradicionais utilizadas no processamento de imagens. As GCNs são projetadas para lidar com dados esparsos, aproveitando a estrutura do grafo para agregar informações dos nós vizinhos. Isso permite capturar relacionamentos complexos e não lineares entre os elementos do grafo. Além disso, as GCNs têm a capacidade de incorporar facilmente informações adicionais na estrutura do grafo, o que as torna versáteis e eficazes em uma variedade de aplicações (ZHANG et al., 2019).

Nas implementações mais simples de uma GCNs, os princípios fundamentais são as agregações de representações de nós a partir da vizinhança (nós vizinhos) e a função de ativação não linear aplicada nessas agregações (ZHANG et al., 2019). Funções de ativação não lineares, como a ReLU *(Rectified Linear Unit)*, são introduzidas na arquitetura porque permitem que a rede neural aprenda e modele propriedades complexas nos dados. Sem o emprego rigoroso dessa não-linearidade, as múltiplas camadas da GCN comportariam-se estritamente como uma transformação linear única, o que incapacitaria o modelo de processar as nuances semânticas do conhecimento extraído. Para a primeira operação, tem-se que para cada nó v no grafo, são agregados os vetores de características dos nós vizinhos, de forma que:

$$h_v^{(1)} = \frac{1}{|N(v)|} \sum u \epsilon N(v)^{x_u} \tag{3.1}$$

, onde $h_v^{(1)}$ é a característica de primeiro nível do nó v, x_u é o recurso de entrada do nó v, e $|N(v)|$ é o número de vizinhos do nó v. Em seguida, a transformação linear aplicada nessas

agregações, é denotada como:

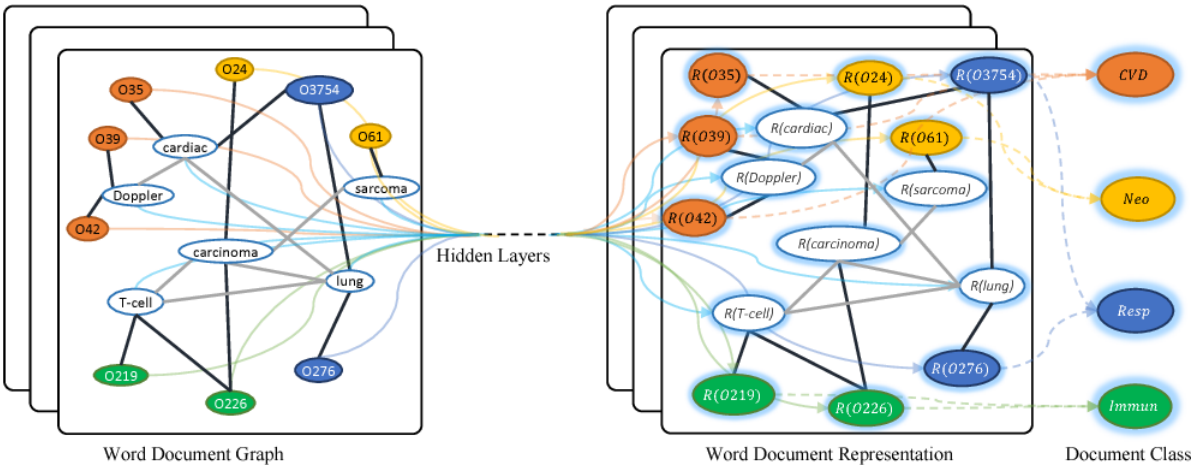
$$h_v^{(2)} = \sigma(W h_v^{(1)}) \tag{3.2}$$

, onde $h_v^{(2)}$ é o segundo nível de recurso do nó v, W é a matriz de pesis, e σ denota a função não-linear de ativação, como a ReLU[3], por exemplo.

As GCNs podem ser aplicadas em diversos domínios como: visão computacional, processamento de linguagem natural, ciência, entre outros. No âmbito da visão computacional, as GCNs têm sido aplicadas para preencher as lacunas existentes nos métodos tradicionais de visão computacional baseados em redes convolucionais clássicas (ZHANG et al., 2019). Em classificação de imagens, por exemplo, algumas abordagens supervisionadas podem converter imagens não estruturadas em dados estruturados em grafos, como o modelo baseado em GCNs proposto por (NARASIMHAN; LAZEBNIK; SCHWING, 2018), no qual utiliza-se um método para processar imagens construindo uma incorporação baseadas em conceitos visuais detectados na imagem para combinar esse conhecimento com conhecimento comum (adquirido pelo usuário).

No campo do processamento de linguagem natural, as GCNs também são utilizadas para classificação de textos, onde os documentos podem ser representados como nós e as relações de citação como arestas. O TextGCN, proposto por (YAO; MAO; LUO, 2019), modela um *corpus* inteiro como um grafo heterogêneo, aprendendo incorporações de palavras e documentos simultaneamente. A Figura 11 ilustra um exemplo do funcionamento do TextGCN. Os nós que começam com "O" representam os documentos, enquanto os demais nós representam palavras. Esse método permite que a coocorrência global de palavras seja claramente modelada e que a convolução de grafos possa ser facilmente ajustada. A soma do número de palavras únicas (tamanho do vocabulário) em um corpus e do número de documentos (tamanho do *corpus*) resulta no número total de nós no grafo de texto (YAO; MAO; LUO, 2019).

Figura 11 – Representação esquemática do TextGCN



Fonte: (YAO; MAO; LUO, 2019)

[3] é uma função de ativação que introduz a propriedade de não linearidade a um modelo de aprendizado profundo.

Além das *Graph Convolutional Networks* (GCNs), destacam-se as *Graph Attention Networks* (GATs), que incorporam mecanismos de atenção no processo de propagação (ZHOU et al., 2020). Esse mecanismo de atenção possibilita que a rede atribua pesos diferentes aos vizinhos de um nó com base em suas características e conexões. Em vez de tratar todos os vizinhos igualmente, como nas GCNs convencionais, as GATs permitem uma personalização dinâmica e adaptativa da importância dos vizinhos durante a agregação de informações, o que pode melhorar significativamente a capacidade da rede de capturar padrões complexos e sutis nos dados de grafos.

A arquitetura desse modelo recebe como entrada uma série de *features* F de nós N, denotado como $h = h_1, h_2, ..., h_N, h_1 \in R^F$, e espera como saída uma série de *features* de diferentes cardinalidades, ou seja, um novo conjunto com diferentes características $F^{'}$. Para essa transformação, é parametrizado uma matriz de peso $W \in R^{F' \times F}$, a cada nó, e em seguida, calculada a importância das *features* dos nós aplicando o método de auto-atenção α, que calcula os coeficientes de atenção, que indicam a importância das características do nó *j* para o nó *i*. O coeficiente de atenção é calculado como:

$$e_i j = \alpha(Wh_i, Wh_j) \tag{3.3}$$

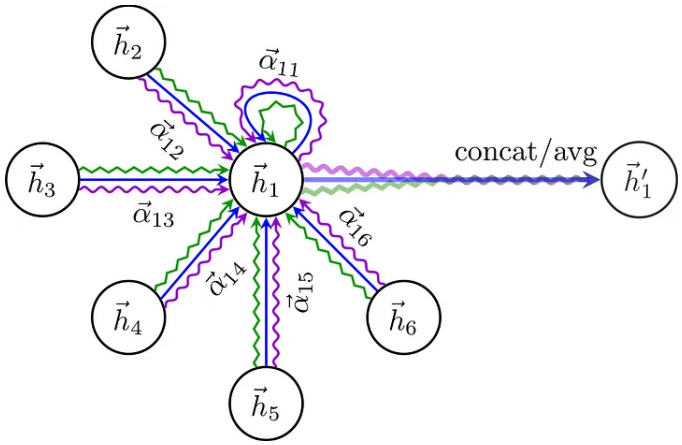
, sendo $\alpha : R^{F'} \times R^{F'} \rightarrow R$.

Na formulação mais geral do modelo, cada nó pode se *atentar* a qualquer outro nó, eliminando todas as informações estruturais. Para incorporar a estrutura do grafo, realizamos atenção mascarada, onde calculamos e_{ij} apenas para nós $j \in N_i$, onde N_i, é algum conjunto de vizinhos do nó i. Além disso, o GAT utiliza a atenção multi-cabeça, como proposto por (VELIčKOVIć et al., 2018), para estabilizar o processo de aprendizagem. Nesse processo, a rede aplica K matrizes de atenção independentes para computar os estados ocultos e, em seguida, concatena suas características (ou calcula a média), conforme demonstrado na Figura 12.

Os experimentos conduzidos por Vaswani evidenciaram que as camadas de atenção utilizadas nas GATs são eficientes computacionalmente, pois não requerem operações matriciais custosas e são paralelizáveis entre todos os nós do grafo. Isso as torna adequadas para lidar com grafos grandes e complexos, onde a computação tradicional poderia se tornar impraticável. Além da eficiência computacional, as GATs demonstraram ser eficazes em contextos tanto de aprendizado transdutivo quanto de aprendizado indutivo. No aprendizado transdutivo, onde o grafo completo está disponível durante tanto o treinamento quanto o teste, as GATs conseguem aproveitar completamente a estrutura do grafo para realizar inferências precisas e generalizadas.

Por outro lado, no aprendizado indutivo, onde os grafos de teste são totalmente novos e não foram vistos durante o treinamento, as GATs conseguem adaptar-se de maneira eficaz, indicando uma boa capacidade de generalização para novos dados após a fase inicial de treinamento.

Figura 12 – Representação do método de atenção “multi-cabeça”



Fonte: (BAO; XU; WANG, 2024)

Outra variação das Redes Neurais em Grafos (GNNs) são as *Graph Recurrent Networks* (GRNs) ou *RecGNNs*, que utilizam arquiteturas neurais recorrentes para aprender representações de nós em grafos. Essencialmente, as GRNs operam sob a suposição de que um nó em um grafo troca continuamente informações com seus vizinhos até que um equilíbrio estável seja alcançado. Essa ideia de troca de mensagens entre nós é inspirada nas redes neurais convolucionais e adaptada para o contexto de grafos, permitindo que as GRNs capturem dinamicamente informações contextuais e relacionamentos entre os nós durante a inferência das representações nodais (WU et al., 2021).

Portanto, observa-se que a estruturação em grafos permite a compreensão e navegação de sistemas complexos a fim de facilitar diversas tarefas de aprendizagem. Todavia, a aplicação prática traz consigo alguns desafios significativos, incluindo complexidade computacional, baixa escalabilidade e dificuldades de integração com métodos tradicionais de aprendizado de máquina. Logo, as GNNs emergem como uma solução promissora para contornar esses impasses. Ao transformar elementos estruturais de grafos em representações vetoriais densas e de baixa dimensionalidade, as GNNs permitem reduzir a complexidade e preservar a informação estrutural, facilitando a aplicação em conjunto de algoritmos de aprendizado de máquina. Pesquisas recentes na literatura científica evidenciam que as GCNs demonstram capacidade de inferir informações topológicas a fim de executar diversas tarefas do mundo real, como predição de tráfego, tarefas de processamento de linguagem natural e sistemas de recomendação. A contínua evolução das metodologias de aprendizado de representação em grafos, incluindo as *Graph Convolutional Networks* (GCNs), *Graph Attention Networks* (GATs) e *Graph Recurrent Networks* (GRNs), sublinha o potencial das GNNs em diversas áreas de aplicação, destacando seu potencial adaptativo e eficiência computacional.

Nesse contexto, este estudo explora a abordagem híbrida da aplicação de Redes Neurais baseadas em Grafos com modelos generativos para a classificação de textos. Além das técnicas de processamento de linguagem natural, também são utilizados conceitos de identificação e

reconhecimento de entidades e contextos semânticos. Os documentos, processados e refinados pelo modelo generativo, são analisados por duas abordagens de extração de conhecimento em triplas (entidade, relação, entidade): uma pelo próprio modelo generativo e outra pelo REBEL (*Relation Extraction By End-to-end Language Generation*) (CABOT; NAVIGLI, 2021), que usa o BERT e um dataset próprio para extrair triplas. Para a etapa de classificação, as representações de conhecimento são processadas por um modelo de Graph Convolutional Networks (GCN), que propaga as informações dos nós através das camadas da rede, enriquecendo a compreensão do conteúdo.

3.4 Classificação de texto

Dentro do campo do Processamento de Linguagem Natural (PLN) e dos modelos de linguagem, destaca-se uma das tarefas mais estudadas e exploradas: a classificação de texto. Essa tarefa envolve a aplicação de técnicas de aprendizado de máquina para extrair categorias ou rótulos de documentos de texto com base em seu conteúdo (KOWSARI et al., 2019). Atualmente, a maioria dos LLMs disponíveis na web é capaz de executar as mais complexas tarefas de classificação de documentos extensos e variados com alta precisão (MINAEE et al., 2024). Além disso, a classificação de texto é utilizada em diversas aplicações do mundo real, como filtragem de *spam* em e-mails, identificação de sentimentos em textos e triagem de conteúdo, entre outras.

Essencialmente, a classificação pode abranger quatro níveis diferentes de escopo: nível de documento, em que o classificador retorna categorias relevantes para o documento como um todo; nível de parágrafo, que obtém categorias de um único parágrafo; nível de frase, que identifica categorias relevantes em uma única sentença; e nível de subfrase, em que o classificador retorna categorias de subexpressões dentro de uma frase.

Algumas das principais técnicas de *feature extraction* mencionadas na literatura são: *n-grams*, que consistem na criação de sequências de n palavras consecutivas para capturar padrões contextuais; *Bag of Words (BoW)*, que representa o texto como um vetor de frequência de palavras, ignorando a ordem e a gramática; *Term Frequency-Inverse Document Frequency (TF-IDF)*, uma técnica estatística que avalia a importância de uma palavra em um documento em relação a uma coleção; e *Word Embeddings*, que geram representações vetoriais densas capazes de captar relações semânticas e sintáticas entre palavras, sendo *Word2Vec*, *GloVe* e *FastText* alguns dos principais exemplos.

Entretanto, esses métodos tem dificuldade de lidar com relações complexas entre palavras e documentos (YAO; MAO; LUO, 2019) e explorar o contexto semântico entre essas conexões. Logo, as GNNS surgem como uma alternativa para esse obstáculo, pois possuem capacidade de explorar as relações semânticas entre os elementos de uma texto de forma global, a nível de *corpus* ou documento.

No que diz respeito à construção de um modelo de classificação, as redes neurais

baseadas em grafos necessitam de um passo a mais logo na escolha do conjunto de dados, pois as informações precisam estar estruturadas em nós e arestas para que o modelo consiga processa-las. Na maioria das aplicações, as estruturas de grafo estão explícitas, porém em tarefas de classificação textual é necessário definir o tipo de abordagem que será utilizado no momento da construção do grafo (WANG; DING; HAN, 2024). Essencialmente, existem dois tipos: nível de *corpus*, em que o grafo representa o *corpus*[4] inteiro. Por outro lado, estruturas a nível de documento focam em representações existem em um único elemento do *corpus*.

Além da construção do grafo, é necessário também definir adequadamente as representações iniciais de nós e arestas para a construção de um modelo de classificação utilizando GNNs. A escolha das representações de nós dependem do tipo de entidade (palavra ou documento) e do tipo de grafo adotado, se forem palavras, os nós podem ser construído utilizando *embeddings* contextuais, como *Word2Vec*, *GloVe* e *FastText*, que têm limitações semânticas, e não contextuais, como *ELMo*, *BERT* e *GPT*, que capturam melhor os significados em diferentes contextos (WANG; DING; HAN, 2024). Se forem a nível de documento, os nós podem ser representações de palavras por meio de redes neurais ou vetores baseados em TF-IDF.

Para as características de arestas deve-se considerar a aprendizagem mais eficaz para capturar as relações explicitas e implícitas entre os nós. Essencialmente, tem-se dois tipos de arestas: estruturais, que possuem relações explicítas como dependências sintáticas, adjacência entre palavras, e não estruturais, que possuem relações implícitas. Em configurações não estruturais, as arestas são comumente definidas través de métodos que atribuem pesos a elas (WANG; DING; HAN, 2024), como *PMI (Point-wise Mutual Information)*, que mede coocorrência de palavras em uma janela deslizante, e *TF-IDF*, usado para pesar as conexões entre nós de nível documento e palavra.

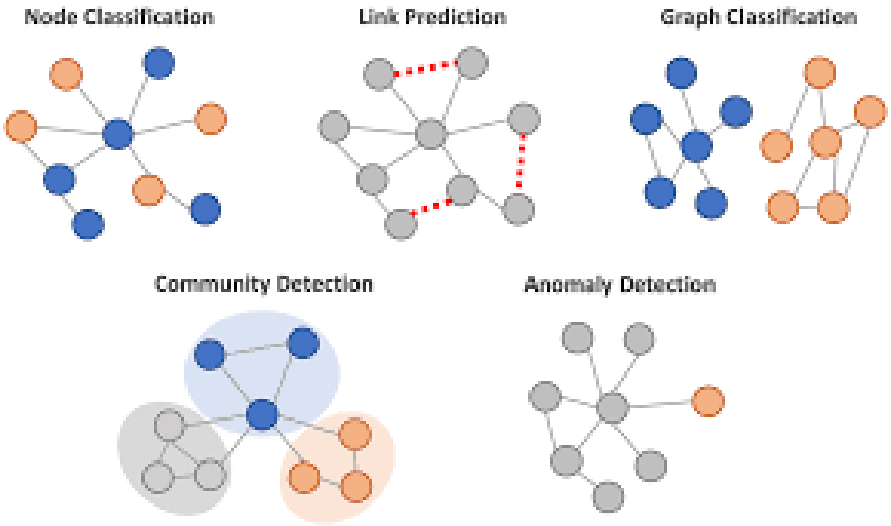
Após a construção do grafo e definição dos nós e arestas, é necessário determina o nível de tarefa de classificação no modelo, que pode ser categorizado em três níveis: nível de nó, nível de aresta e nível de grafo.

- Tarefas a nível de nós: Tarefas focadas nos objetos/entidades, como *node classification* (classificação de nós), que visa categorizar os nós em classes; *node regression* (regressão em nós), focado na predição de valores contínuos para cada nó, e *node clustering* (clusterização de nós), que visa agrupar nós similares em grupos relacionados;
- Tarefas a nível de aresta: Tarefas focadas na conexão/relacionamento entre nós, como *edge classification* (classificação de arestas) e *link prediction*, no qual o modelo classifica os tipos de arestas ou prediz se existe uma conexão/relacionamento (aresta) em dois nós predefinidos (WU et al., 2021);
- Tarefas a nível de grafo: Tarefas como *graph classification*, na qual o modelo precisa aprender as representações do grafo;

[4] Representa um coleção de documentos

A Figura 13 complementa a categorização de tarefas comuns que a GNN pode ser aplicada. A primeira tarefa, é a *node classification*, que, conforme citado anteriormente, envolve a atribuição de rótulos a nós individuais com base em suas características e conexões. *Link prediction* (Previsão de Links), que refere-se à previsão de conexões futuras ou ausentes entre nós, útil em redes sociais e biológicas. *Graph classification* (Classificação de Grafos), que consiste em classificar grafos com base em suas propriedades estruturais, aplicável em química e bioinformática. E, adicionalmente, mais dois tipos de tarefas: *community detection* (detecção de comunidades), que identifica grupos de nós densamente conectados, revelando estruturas subjacentes em redes; e, *anomaly detection* (detecção de anomalias), que foca na identificação de nós ou arestas que desviam do padrão esperado, crucial para segurança e monitoramento.

Figura 13 – Representação visual de diferentes aplicações da GNN



Fonte: Elaborado pelo autor

A respeito do segundo critério, estabelecem-se três possíveis configurações de treinamento (ZHOU et al., 2020):

- Supervisionado:os dados rotulados são fornecidos, permitindo treinar modelos para tarefas específicas, como classificação de nós ou de grafos inteiros;
- Semi-supervisionado: utiliza uma pequena quantidade de dados rotulados em conjunto de dados não rotulados. É uma abordagem eficiente para melhorar a precisão das previsões. As *ConvGNNs* (Redes Neurais Convolucionais de Grafos) são comumente empregadas nesse contexto, utilizando camadas convolucionais de grafos seguidas por uma camada *softmax* para a classificação multiclasse, permitindo inferir os rótulos dos nós desconhecidos (WU et al., 2021);
- Não-supervisionado: apenas dados não rotulados são utilizados. A clusterização de nós é uma típica tarefa de aprendizado não supervisionado, onde o objetivo é agrupar nós semelhantes sem a necessidade de rótulos;

A etapa de redução de dimensionalidade visa lidar com conjuntos de dados textuais que contém muitas palavras únicas, reduzindo os problemas de complexidade computacional e memórias durante as etapas de pré-processamento, pois reduz o tamanho do espaço de características em modelos vetoriais baseados em termos (KOWSARI et al., 2019). Algumas das técnicas mais populares encontradas na literatura são:

- *Principal Component Analysis (PCA)*: que identifica subespaços para maximizar a variabilidade dos dados e pode ser usado para pré-processamento e redução de ruído.
- *Independent Component Analysis (ICA)*: método de modelagem estatístico no qual os dados observados são representados como transformações lineares de componentes independentes.
- *Linear Discriminant Analysis (LDA)*: uma técnica que separa múltiplas classes com múltiplas características, dentificando uma combinação linear dessas características para categorizá-las em suas respectivas classes.
- *Non-Negative Matrix Factorization (NMF)*: se mostra poderosa para dados de alta dimensionalidade, como textos, ao decompor matrizes em componentes menores.
- Projeções aleatórias, incluindo as técnicas de *Random Kitchen Sinks e o Johnson Lindenstrauss Lemma*: projeções aleatórias agrupam técnicas bastante utilizadas em alto volume de dados e em tarefas como mineração de texto, classificação de texto, dentre outras.
- *Autoencoders*: são um tipo de redes neurais projetadas para copiar entradas para saídas, sendo eficazes na redução de dimensionalidade pela representação compacta de dados complexos. O ponto-chave é que entre as camadas de entrada e saída, exista uma camada oculta com menos unidades que pode ser usada para reduzir as dimensões de um espaço de características.
- *t-SNE*: é uma técnica não-linear voltada para visualização de dados em espaços de baixa dimensionalidade, preservando a similaridade local entre pontos de dados.

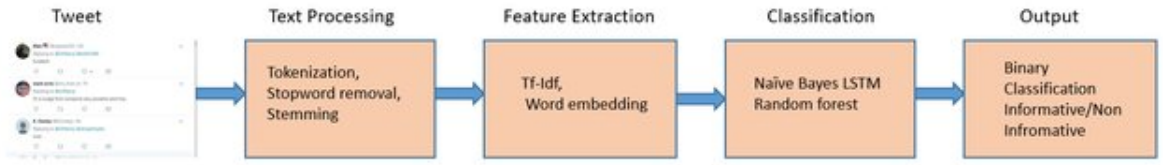
A terceira etapa no pipeline de classificação de textos é a escolha do classificador. Essa etapa se torna efetiva, uma vez que o escopo do problema é bem claro, para que assim possa determinar o melhor algoritmo de classificação. Dentre os diversos modelos existentes, alguns se destacam na comunidade científica, tais como: *Support vector machines (SVMs), Naïve Bayes, decision tree, random forest*, e outras abordagens baseadas em redes neurais, como: *deep neural networks (DNN), CNN, RNN*, dentre outras (KOWSARI et al., 2019). Além disso, é comum que muitos pesquisadores combinem uma ou mais técnicas de diferentes tipos a fim de obter um modelo de classificação mais robusto e preciso.

A quarta e última etapa no pipeline é a validação, em que o desempenho e resultados do modelo são submetidos a uma série de avaliações para medir a sua eficácia. No entanto, a avaliação de modelos de classificação de texto enfrenta desafios devido à falta de um padrão específico e protocolo para garantir consistência nos resultados (KOWSARI et al., 2019). Para

contornar esse problema, é essencial utilizar uma variedade de métricas que abrangem diversos critérios relacionados à performance, a fim de garantir a comparabilidade dos resultados.

A Figura 14 ilustra um exemplo completo do pipeline de classificação de texto. Começando com um conjunto de dados, como tweets no exemplo dado, são aplicadas etapas sequenciais de processamento de texto, extração de características e classificação. O objetivo final é atribuir categorias ou rótulos relevantes aos textos no conjunto de dados.

Figura 14 – Representação visual do pipeline da tarefa de classificação textual



Fonte: (GAUTAM; KUMAR; SHAH, 2019)

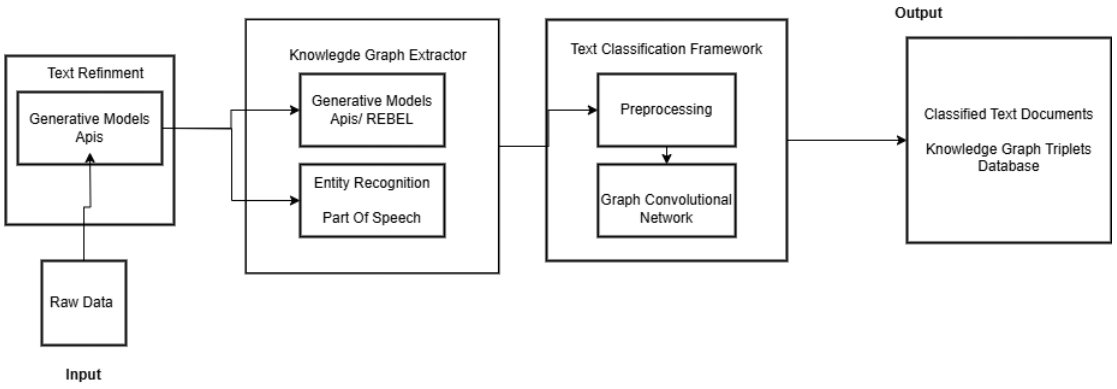
4 Metodologia

Este capítulo detalha o desenvolvimento do framework proposto. O foco central abordado pelo modelo visa sanar a dificuldade das Redes Neurais de Grafos convencionais em extrair e aplicar conexões factuais explícitas (semântica profunda) sem incorrer nos gargalos de altíssima complexidade inerentes aos grafos modernos. A solução apresentada constitui uma arquitetura híbrida que emprega Large Language Models (Gemini) e arquiteturas seq2seq (REBEL) para refinar e extrair conhecimento em triplas estruturadas de relacionamento. Estas representações estruturadas são, então, combinadas a atributos lexicais em um grafo heterogêneo e processadas por uma Graph Convolutional Network (GCN) supervisionada para executar a classificação final dos textos documentais.

4.1 Framework GenGraph

A Figura 15 exibe as etapas principais do framework proposto, começando pela etapa de refinamento de texto utilizando o Gemini Api, partindo para a extração de conhecimento utilizando o Gemini Api e REBEL, seguindo para a representação em grafo das triplas extraídas, e, por fim, utilizando um *Graph Convolutional Network* para executar tarefas de classificação a partir das representações grafais aprendidas.

Figura 15 – Fluxograma do framework proposto



Fonte: Elaborado pelo autor

4.2 Bases de Dados Selecionadas

A coleta de dados consistiu na seleção e armazenamento de bases textuais em repositórios abertos, sendo estes: AG News Subset, 20 Newgroups, Reuters Dataset e BBC News (PT-Br). O primeiro conjunto contém uma coleção de mais de um milhão de artigos de notícias acadêmicas e foi extraído através do catálogo aberto de *datasets* disponível pelo TensorFlow, que reúne várias coleções de dados disponíveis para serem usadas em pesquisas e estudos acadêmicos. O conjunto

possui 3 colunas (*description, label e title*), das quais foram utilizadas as informações da coluna de *description*, que armazena os textos completos das notícias e a coluna *label*, que armazena as categorias das notícias, que são: ciência/tecnologia, esportes, negócios e mundo. A segunda fonte de dados é uma coleção de mais de vinte mil documentos de 20 tópicos distintos, que foi popularizada na comunidade científica para ser utilizada durante o treinamento de modelos de aprendizado de máquina.

O Reuters Dataset, contém mais de dez mil documentos de notícias de tópicos diversos, e é uma das coleções mais amplamente utilizadas de artigos da agência de notícias financeiras Reuters. É um benchmark essencial para pesquisas em categorização de texto. A coleção inclui colunas com informações detalhadas, como o texto completo do artigo (*text*), o tipo de texto (*text-type*), indicando se pertence ao conjunto de treino ou teste, e os tópicos associados ao documento (*topics*), além de locais, pessoas, organizações e bolsas mencionadas, data e título.

Por fim, utilizou-se o dataset BBC News (PT-Br), uma coleção que reúne diversas notícias de categorias distintas no site da BBC no idioma Português Brasileiro. A coleção inclui as colunas de categoria da notícia, título, texto completo, data de publicação e link para a notícia no site oficial da BBC. Para esse trabalho, utilizou-se as colunas de categoria e texto.

A diversidade de categorias de documentos textuais e do idioma utilizado fornece uma base sólida para o pré-treinamento do modelo. A tabela 1 exibe detalhes adicionais a respeito do tamanho, número de tokens e classes dos conjuntos de dados escolhidos.

Tabela 1 – Informações gerais dos conjuntos de dados

Dataset	**Documentos**	**Tokens**	**Classes**
Reuters	10.788	29.930	90
AG News	127.600	7.600	4
20 Newsgroups	18.000	130.107	20
BBC News	8637	118.714	9

Fonte: Elaborada pelo autor.

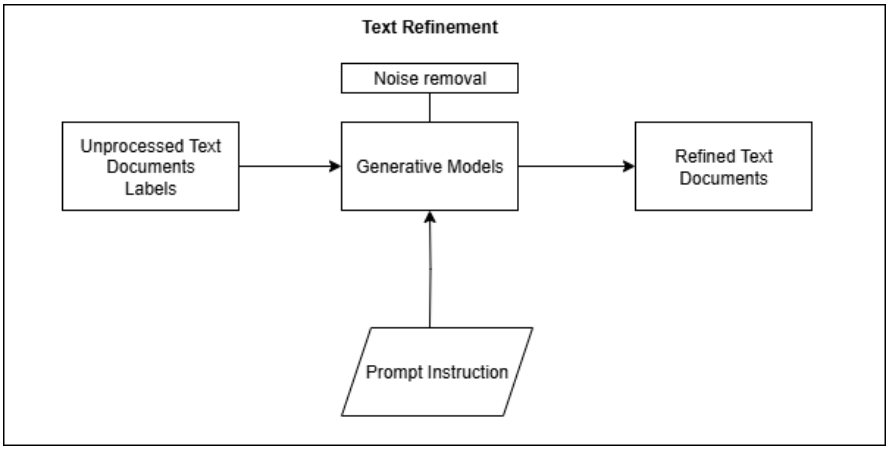
4.3 Pré-processamento de dados com Gemini AI

Nesta etapa, foi utilizada a API do Gemini AI para refinar documentos textuais dos datasets selecionados. Para isso, foi elaborado um *prompt* que define os critérios a serem seguidos pelo modelo, como a correção de erros gramaticais e ortográficos, mantendo o sentido semântico original do texto. A construção do prompt baseou-se no estudo de *TextGraph* (SHI et al., 2023), que também empregou um modelo generativo para refinamento textual. O objetivo da instrução foi corrigir erros ortográficos e aprimorar a compreensão do texto original.

A Figura 16 detalha as etapas e técnicas aplicadas nessa etapa, os documentos não processados são enviado para o modelo generativo, que, através de uma instrução *prompt* aplica técnicas de processamento de dados, incluindo *noise removal*, que é a remoção de ruido, ou seja,

a remoção de componentes textuais ou não que possam prejudicar a tarefa de classificação e que podem ser removidos sem alterar a semântica do texto.

Figura 16 – Fluxograma do módulo de refinamento de texto



Fonte: Elaborado pelo autor

O *prompt* inclui instruções adicionais relacionadas ao critério de abstração, destacando a necessidade de preservar o sentido original do texto e enfatizando a proibição de adicionar, remover ou modificar informações presentes no conteúdo original. A Figura 17 exibe o prompt desenvolvido para a execução desta etapa.

Figura 17 – Prompt de refinamento de texto

```
Please generate a refined document of the
following document. And please ensure that
the refined document meets the following
criteria:
1. Ensure the refined version remains abstract and does
not change the original meaning.
2. Retain all important elements, concepts, and
relationships from the original document.
3. Do not add, remove, or modify information beyond
what exists in the original content.
4. The refined version must be clear, concise, and easy
to read, with no abbreviations or spelling errors.
5. Present the content in a structured, organized
format for better readability.
Here is the content to refine: [content]
```

Fonte: Elaborado pelo autor

Além de definir os critérios de refinamento, são realizadas configurações adicionais na API do Gemini para permitir o processamento de conteúdos potencialmente sensíveis. Por fim, todas as versões refinadas e processadas dos documentos são armazenadas em arquivos CSV separados, os quais serão utilizados nas etapas subsequentes do *framework* proposto.

4.4 Extração de conhecimento

A etapa de extração de conhecimento consistiu em converter os documentos refinados pelo Gemini em estruturas textuais de grafos de conhecimento em representação de triplas, as quais contêm uma coleção de fatos que podem ser denotadas como: $(h, r, t) \subseteq \varepsilon \times R \times \varepsilon$, onde ε e R definem as entidades e as relações entre si. A extração se deu por duas abordagens distintas: 1) Utilização do Gemini Api para extrair conhecimento a partir de um *prompt* pré-definido e, 2) Extração de conhecimento utilizando *REBEL*, um modelo baseado em BERT que identifica entidades e relacionamentos no texto.

Para a primeira abordagem, foi definido um prompt que definia os seguintes critérios para extração de conhecimento: identificação de entidades no texto, identificação dos relacionamentos e conexões entre as entidades, resumo dos relacionamentos da forma mais curta e coesa possível, e o *output* no formato padrão de *('head entity', 'relation', 'tail entity')*, ou seja, *('entidade', 'relacionamento', 'entidade')*. E, no caso de não existir um resultado válido para a instrução, retornar apenas *None*.

O formato de identificação de entidades escolhido foram os seguintes: *Named entity recognition (NER)*, uma técnica dentro da área da NLP que identifica elementos textuais e classifica-os em categorias pré-determinadas como nomes de pessoas, organizações, locais, data, valores monetários, dentre outros. E *Part-of-Speech* (POS), outra técnica da NLP que designa uma categoria gramatical para cada elemento textual identificado, como substantivo, verbo, adjetivo, etc, com o objetivo de entender a estrutura gramatical e semântica do texto.

A segunda abordagem utiliza o REBEL, ou, *Relation extraction by end-to-end language generation*, um modelo que simplifica a tarefa de extrações de entidades e relações, utilizando o método de *seq2seq* ou *sequence-to-sequence*, um modelo de arquitetura que transforma um elemento sequencial em outro, muito utilizado em tarefas de tradução ou geração textual. Essencialmente, o *REBEL* recebe um elemento textual de entrada e gera uma sequência de triplas que denotam explicitamente as relações entre as entidades no texto. Além disso, consegue extrair mais de 200 tipos de relações, e possui uma integração com a base de dados da *Wikipedia*, o que potencializa a tarefa de extração de entidades (CABOT; NAVIGLI, 2021).

A Figura 18 mostra o *prompt* desenvolvido para a execução da primeira abordagem utilizando a api do Gemini.

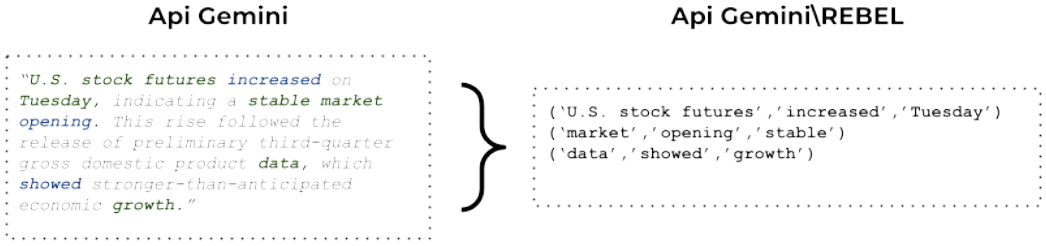
Figura 18 – Prompt de extração de conhecimento

```
You are a knowledge graph extractor, and your task is to extract and return
a knowledge graph from a given text. Let's extract it step by step:
(1). Identify the entities in the text. An entity can be a noun or a noun
phrase that refers to a real-world object or an abstract concept. You can
use a named entity recognition (NER) tool or a part-of-speech (POS) tagger
to identify the entities. No need to return the answer for this step.

(2). Identify the relationships between the entities. A relationship can be
a verb or a prepositional phrase that connects two entities. You can use
dependency parsing to identify the relationships. No need to return the
answer for this step.

(3). Summarize each entity and relation as short as possible and remove any
stop words. No need to return the answer for this step.

(4). The response of the knowledge graph must be the triplet format given
below: ('head entity', 'relation', 'tail entity').

(5). Most importantly, if you cannot find any knowledge, please just
output: 'None'.
Here is the content: [content]
```

Fonte: Elaborado pelo autor

A Figura 19 exibe um exemplo do processo de extração de conhecimento feito pelo Gemini e REBEL, onde cada ferramenta recebe o texto-base já refinado e extrai as entidades e relacionamentos do formato definido padrão mencionado anteriormente. No exemplo, as palavras em verde representam as entidades e as em azul representam o relacionamento entre elas.

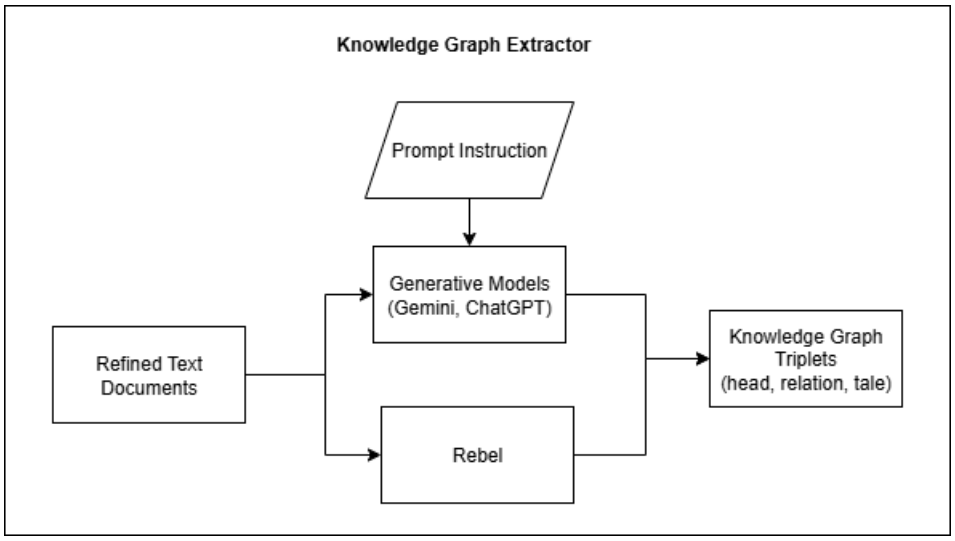
Figura 19 – Exemplo do processo de extração de conhecimento



Fonte: Elaborado pelo autor

A Figura 20 apresenta um fluxograma das etapas descritas anteriormente, apresentando as duas abordagens utilizadas e a saída esperada ao final da execução dessa etapa.

Figura 20 – Fluxograma do módulo de extração de conhecimento



Fonte: Elaborado pelo autor

4.5 Classificação Textual com GCN

Esta etapa consistiu no desenvolvimento de uma *Graph Convolutional Network (GCN)* para realizar classificação textual aplicando *node classification*. *Node classification* é uma tarefa que tem como objetivo designar rótulos aos nós do grafo, baseados nas propriedades dos nós e as relações entre eles. Essencialmente, a construção do modelo se deu através das seguintes etapas: representação do grafo e modelagem da arquitetura de rede.

4.5.1 Carregamento dos dados

Após passar pelas etapas descritas anteriormente, as triplas de conhecimento extraídas na seção 4.3 são armazenadas em um dataset contendo as colunas *head*, *relation*, *tail*, *label* e *docid*, ou seja, as triplas extraídas de cada documento, o label/rótulo relacionado e o id do documento.

4.5.2 Representação do grafo

Esta etapa diz respeito à estruturação dos dados de *input* de forma que pudessem ser processados pelo modelo, ou seja, o conhecimento extraído na etapa anterior é transformado em uma representação gráfica, de forma que possa ser processada pelo modelo GCN.

A estruturação de um grafo, para que seja possível ser manipulado por um modelo neural, necessita de três componentes-chave: a) matriz de *features* de nós: uma matriz que contém informações de todos os nós do grafo, e é representada como X e tem o formato de número de nós $\times$ número de *features*; b) *edge index*: um tensor que define as conexões entre os nós; e c) rótulos de nós: uma matriz que armazena os rótulos de cada nó e tem o formato de número de nós $\times$ número de classes, onde cada linha corresponde a um nó e cada coluna a um rótulo.

Partindo da análise inicial das bases de dados escolhidas, é possível identificar que os grafos resultantes são do tipo dirigidos, pois as relações possuem uma direção "direta"; heterogêneos, os nós apresentam entidades distintas e as arestas possuem diferentes tipos de relação, e são estáticos, pois os dados não indicam mudanças ao longo do tempo.

Com as triplas semânticas extraídas dos documentos, as fases subsequentes concentram-se em estruturar esses dados em um formato compatível com o modelo. Esse processo é fundamental e envolvem diversas subetapas, iniciando-se com a construção do *dataframe* do documento, bem como construção de arestas, matrizes de *features*, até a subetapa final da construção do grafo.

- **Construção do Dataframe e Codificação de Rótulos:** nessa subetapa, as triplas extraídas são agrupadas por *id* (identificador) do documento e em seguida, associadas aos seus respectivos rótulos. Visto que modelos de aprendizado de máquina requerem entradas numéricas, estes rótulos categoriais (e.g., *strings*) são convertidos para uma representação numérica. Para esta finalidade, emprega-se a classe `LabelEncoder` da biblioteca *scikit-learn*, que permite a tranformação de labels não-numéricas em númericas (valor inteiro).
- **Definição da Estrutura de Arestas (*Edges*):** são definidas as conexões (arestas) que estruturam o grafo através de duas categorias principais: arestas Documento-Palavra e Entidade-Entidade. Para as arestas Documento-Palavra, cada documento é conectado bidirecionalmente às suas palavras mais relevantes, identificadas através dos scores TF-IDF mais altos (top-20 palavras por documento). Essas conexões são representadas em um tensor com formato (2, $N_{\text{doc-word}}$), onde a primeira linha contém os índices dos nós de origem (documentos) e a segunda, os de destino (palavras). Para as arestas Entidade-Entidade, os nós são conectados com base nas relações semânticas extraídas das triplas do grafo de conhecimento no formato (*subject* $\rightarrow$ *relation* $\rightarrow$ *object*). Estas conexões bidirecionais são consolidadas em um tensor de formato (2, $N_{\text{ent-ent}}$), sendo posteriormente convertidas para formato não-direcionado.
- **Construção das matrizes de (*features*):** Com a estrutura topológica do grafo definida pelas arestas, a próxima etapa consiste em gerar as representações vetoriais (características) para cada nó. Para isso, utiliza-se vetorização *TF-IDF* (*Term Frequency-Inverse Document Frequency*) através da classe `TfidfVectorizer` da biblioteca *scikit-learn*, que permite capturar a importância relativa das palavras nos documentos. Reconhece-se que métodos tradicionais baseados em frequência, como o TF-IDF, ignoram o contexto sequencial sintático e a ordem gramatical, essencialmente perdendo o direcionamento semântico rico que vetores densos gerados por embeddings contextuais (como Word2Vec, BERT ou GPT) costumam capturar de maneira superior. Contudo, a escolha técnica pelo TF-IDF justifica-se para esta configuração específica devido à sua alta eficiência em cenários macro e de nível de documento. Visto que a topologia relacional robusta (a conexão semântica) já está intrinsecamente mapeada

pelas arestas do grafo de conhecimento recém-extraído (Entidade-Entidade), o uso dos vetores de características baseados em TF-IDF cumpre adequadamente o papel de fixar as importâncias lexicais estatísticas de cada documento na matriz, mitigando a extrema complexidade dimensional que *embeddings* pesados adicionariam sem ganho computacional proporcional.

O vocabulário é limitado às palavras mais frequentes, e cada documento é representado através de seus scores TF-IDF mais significativos. As características dos nós são codificadas como identificadores simples: nós de palavras do vocabulário são indexados pelos seus identificadores no vocabulário TF-IDF, nós de entidades do grafo de conhecimento são mapeados sequencialmente após o vocabulário, e nós de documentos são representados por um token especial. Ao final deste processo, obtém-se uma matriz de características, onde cada linha corresponde a um identificador que representa um nó específico no grafo, seja ele uma palavra, uma entidade ou um documento.

- **Construção do grafo:** A subetapa final, que precede o treinamento do modelo, consiste na criação de um objeto que representa o grafo para manipulação. Para isso, emprega-se a classe `Data` da biblioteca **PyTorch Geometric**, estrutura padrão para representação de grafos homogêneos. Esta classe permite a manipulação das características dos nós e suas conexões, além de prover as funcionalidades básicas de tensores do PyTorch. O grafo heterogêneo resultante contém três tipos de nós: nós de palavras do vocabulário, nós de entidades do grafo de conhecimento, e nós de documentos. As conexões estabelecidas são bidirecionais e incluem tanto as relações documento-palavra (baseadas em scores TF-IDF) quanto as relações entidade-entidade (derivadas das triplas extraídas). Por fim, para cada um dos nós são atribuídas as características (*features*) geradas na etapa anterior através dos identificadores únicos, completando a estrutura do grafo heterogêneo.

4.5.3 Definição da arquitetura da rede

Essencialmente, a construção de uma rede neural concentra-se em definir as camadas e funções de ativação.

O modelo proposto nesta pesquisa utiliza uma arquitetura de Rede Neural Convolucional em Grafos (*Graph Convolutional Network* - GCN) composta por duas camadas de convolução gráfica. As camadas convolucionais em grafos agregam informações dos vizinhos de cada nó através da estrutura do grafo, permitindo que a representação de cada nó seja enriquecida com informações contextuais de sua vizinhança local.

Antes de detalhar a arquitetura, é importante definir os componentes técnicos utilizados para garantir estabilidade e eficácia no treinamento:

- ***Batch Normalization*** (normalização em lote): técnica que normaliza as ativações de cada camada, estabilizando o treinamento, acelerando a convergência e permitindo o uso de taxas de aprendizado mais elevadas.
- ***Dropout***: técnica de regularização que desativa aleatoriamente neurônios durante o treinamento para prevenir o *overfitting*, situação em que um modelo se ajusta excessivamente aos dados de treinamento, perdendo a capacidade de generalizar para novos dados.
- **Conexões residuais** (*skip connections*): conexões diretas que permitem que a informação flua diretamente entre camadas não adjacentes, facilitando o treinamento de redes mais profundas e mitigando o problema de degradação de gradiente.

Com esses componentes definidos, a arquitetura do modelo é composta pelas seguintes camadas:

- **Primeira camada (GCNConv)**: realiza a convolução gráfica nas *features* de entrada, projetando-as para um espaço latente de representação com dimensionalidade reduzida. A dimensionalidade de entrada varia conforme o tipo de representação textual utilizada: embeddings BERT densos (tipicamente 384 ou 768 dimensões, capturando semântica contextual) ou representação TF-IDF esparsa (dimensão configurável baseada no tamanho do vocabulário). Após a operação de convolução, aplica-se sequencialmente: normalização em lote, função de ativação ReLU (para introduzir não-linearidade) e *dropout* (para regularização). Adicionalmente, uma conexão residual é implementada através de uma projeção linear quando a dimensionalidade de entrada difere da dimensão oculta, permitindo que a informação original flua diretamente para camadas posteriores.
- **Segunda camada (GCNConv)**: realiza uma segunda operação de convolução gráfica, mapeando as representações intermediárias para o espaço de saída correspondente ao número de classes do problema de classificação. Esta camada não utiliza função de ativação, produzindo *logits* brutos (valores reais não normalizados) que são posteriormente processados pela função de perda.

Funções de Ativação

As funções de ativação são elementos essenciais na construção de redes neurais, pois introduzem não-linearidade no modelo e permitem o aprendizado de padrões complexos nos dados. Na primeira camada do modelo, utiliza-se a função ReLU (*Rectified Linear Unit*), definida como:

$$\text{ReLU}(x) = \max(0, x) \tag{4.1}$$

onde x é qualquer número real. A função retorna 0 se $x \leq 0$, caso contrário retorna x. A ReLU é amplamente utilizada por sua simplicidade computacional e eficácia em evitar o problema de des-

vanecimento de gradiente durante o treinamento.

Função de Perda

A camada de saída não utiliza função de ativação, produzindo diretamente os *logits* para cada classe. Esses valores brutos são processados pela função de perda *Focal Loss*, que internamente aplica a função *softmax* para converter os *logits* em probabilidades. A função *softmax* é definida como:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{4.2}$$

onde x_i representa o *logit* da classe i e K é o número total de classes. A saída da *softmax* pode ser interpretada como a probabilidade de um nó pertencer a cada uma das classes. Por exemplo, se a saída for 0.7 para um determinado rótulo, isso pode ser interpretado como uma probabilidade de 70% de que o nó pertence a essa classe.

A *Focal Loss*, uma variação da *CrossEntropyLoss* proposta por (LIN et al., 2018a), foi especificamente escolhida para lidar com o desbalanceamento de classes frequentemente presente em datasets de classificação de textos. A *Focal Loss* é definida como:

$$\text{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{4.3}$$

onde p_t é a probabilidade predita para a classe verdadeira e $\gamma \geq 0$ é o parâmetro de foco modulador. O termo $(1 - p_t)^{\gamma}$ atua como um fator de ponderação que reduz a contribuição de exemplos fáceis de classificar (alta confiança) e aumenta a importância relativa de exemplos difíceis (baixa confiança). Quando $\gamma = 0$, a *Focal Loss* é equivalente à *CrossEntropyLoss* convencional. Adicionalmente, pesos de classe podem ser aplicados para balancear a contribuição de classes majoritárias e minoritárias durante o treinamento, tipicamente através de transformações como a raiz cúbica dos pesos balanceados para suavizar o efeito de classes extremamente raras.

4.6 Treinamento da GCN

Para os testes preliminares e treinamento da rede, foi utilizada uma abordagem supervisionada, utilizando dados rotulados para o treinamento do modelo. O processo de treinamento foi configurado com 300 épocas, com monitoramento periódico das métricas de desempenho.

Nesta etapa, foi utilizado o *Adam Optimizer (Adaptive Moment Estimation)*, um algoritmo de otimização amplamente empregado em redes neurais devido à sua eficiência e robustez. O Adam combina as vantagens de dois métodos populares: o Gradient Descent with Momentum e o RMSProp. Essa combinação permite que o Adam adapte dinamicamente as taxas de aprendizado para cada parâmetro do modelo, tornando-o particularmente eficaz em problemas com grandes

volumes de dados ou parâmetros. A taxa de aprendizado inicial foi configurada em 0.003, com decaimento de peso (*weight decay*) de 1×10^{-4} para regularização adicional.

Para garantir a estabilidade do treinamento, especialmente em grafos esparsos, implementa-se *gradient clipping*, que limita a norma dos gradientes a um valor máximo predefinido (1.0), prevenindo explosões de gradiente. Adicionalmente, utiliza-se um agendador de taxa de aprendizado do tipo *ReduceLROnPlateau*, que monitora a métrica de F1-score no conjunto de validação e reduz a taxa de aprendizado em 50% quando o desempenho do modelo estagna por 15 épocas consecutivas, permitindo que o modelo faça ajustes mais finos à medida que se aproxima de um mínimo local. A taxa de aprendizado mínima foi estabelecida em 1×10^{-6}.

Como técnica de regularização, foi implementado *dropout* com taxa de 0.5, aplicado após as operações de convolução gráfica. O *dropout* previne o *overfitting* desativando aleatoriamente 50% dos neurônios durante o treinamento, forçando o modelo a aprender representações mais robustas e generalizáveis. Adicionalmente, foi aplicado *edge dropout* com taxa de 0.15, que remove aleatoriamente 15% das arestas do grafo durante o treinamento, aumentando a robustez do modelo a variações na estrutura do grafo. Ainda como técnica de regularização, foi aplicado *label smoothing* com fator de 0.1, que suaviza os rótulos alvo para prevenir predições excessivamente confiantes e melhorar a generalização.

Já a função de perda escolhida foi a *Focal Loss*, uma variação da *CrossEntropyLoss* especificamente projetada para lidar com desbalanceamento severo de classes. A *Focal Loss* é definida como:

$$\text{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{4.4}$$

onde p_t é a probabilidade predita para a classe verdadeira e $\gamma = 2.0$ é o parâmetro de foco modulador. O termo $(1 - p_t)^{\gamma}$ reduz a contribuição de exemplos fáceis de classificar (alta confiança) e aumenta o peso relativo de exemplos difíceis (baixa confiança), permitindo que o modelo concentre seu aprendizado nas amostras mais desafiadoras. Internamente, a *Focal Loss* aplica a função *softmax* aos *logits* de saída do modelo para obter as probabilidades p_t.

Adicionalmente, pesos de classe podem ser aplicados para balancear a contribuição de classes majoritárias e minoritárias. Estes pesos são calculados utilizando a estratégia *balanced* e posteriormente suavizados através da raiz cúbica para evitar penalizações excessivas em classes extremamente raras. Como alternativa, o modelo também suporta o uso da função *CrossEntropyLoss* convencional, definida como:

$$\text{CE}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{4.5}$$

onde C representa o número de classes, y os rótulos verdadeiros e $\hat{y}$ as probabilidades preditas. A *CrossEntropyLoss* combina a aplicação da função *softmax* com o cálculo da perda de entropia cruzada em uma única operação, proporcionando maior eficiência computacional e estabilidade numérica.

Além disso, as funções de perda são derivadas do conceito de entropia na teoria da informação, que é uma métrica de incerteza associada a uma distribuição de probabilidades. A escolha da *Focal Loss* como função principal foi motivada por sua adequação a tarefas de classificação multiclasse com desbalanceamento extremo de classes, como frequentemente observado em datasets de classificação de textos. O uso combinado com o otimizador Adam, taxa de aprendizado adaptativa e múltiplas técnicas de regularização garantiu um processo de treinamento estável e eficaz, contribuindo para a convergência do modelo.

O treinamento foi estruturado com monitoramento periódico das métricas de desempenho a cada 10 épocas, permitindo acompanhar a evolução da acurácia, F1-score e perda nos conjuntos de treino e validação. Devido à natureza heterogênea do grafo, o cálculo da função de perda foi restrito apenas aos nós que representam documentos, utilizando máscaras específicas para excluir nós de palavras e entidades do processo de otimização direta. O melhor modelo foi selecionado com base na acurácia do conjunto de validação, com seu estado salvo para posterior avaliação no conjunto de teste.

Tabela 2 – Resumo dos hiperparâmetros utilizados no modelo GCN

Hiperparâmetro	**Valor Utilizado**
Épocas (Máximo)	300
Paciência (*Early Stopping*)	50 épocas
Taxa de Aprendizagem (Inicial)	0,01
Weight Decay	1×10^{-4}
Limite de Gradiente (*Clipping*)	1,0
Dropout	0,5
Edge Dropout	0,15
Label Smoothing	0,1

Fonte: Elaborada pelo autor.

4.7 Validação do modelo

A validação é uma etapa crucial do desenvolvimento de modelos de aprendizado de máquina, pois permite avaliar a performance do modelo e verificar se está generalizando adequadamente para dados não vistos durante o treinamento. No contexto das *Graph Convolutional Networks* (GCNs), esta etapa é fundamental para determinar se o modelo aprendeu representações nodais significativas a partir da estrutura do grafo e das características dos nós.

O processo de validação implementado utiliza uma estratégia de *early stopping* baseando-se na acurácia do conjunto de validação para evitar *overfitting*. Embora parte dos *datasets* utilizados (como o Reuters) possua um elevado desbalanceamento de classes, a opção por priorizar a acurácia como métrica primária de estabilidade no monitoramento apoia-se no fato de que o desequilíbrio já estava sendo ativamente tratado via manipulação das funções de perda — especificamente por meio da *Focal Loss* com distribuição balanceada de pesos de classe. Desta

forma, garantir um monitoramento de convergência global sem oscilações abruptas permitiu um ajuste refinado dos gradientes durante as iterações de parada.

Durante o treinamento, o estado do modelo com melhor desempenho de validação é salvo, sendo posteriormente restaurado ao final do processo para garantir que o modelo final seja aquele com melhor capacidade de generalização.

Para quantificar o desempenho, são utilizadas métricas de classificação padrão implementadas através da biblioteca *scikit-learn*. As métricas principais aplicadas no framework são:

- **Acurácia**: A porcentagem de rótulos previstos corretamente em relação ao total de amostras, calculada como $\frac{\text{predições corretas}}{\text{total de amostras}}$.
- **Precisão (Macro)**: A proporção de predições positivas que estavam corretas para cada classe, importante para avaliar a confiabilidade das predições. A média macro trata todas as classes com igual importância.
- **Recall - Sensibilidade (Macro)**: A proporção de positivos reais que foram corretamente identificados pelo modelo, indicando a capacidade de detectar todas as instâncias relevantes de cada classe.
- **F1-Score**: A média harmônica entre precisão e recall, calculada como $\frac{2\times\text{Precisão}\times\text{Recall}}{\text{Precisão}+\text{Recall}}$. O modelo computa três variantes desta métrica:
  - *F1-Score Macro*: Média aritmética do F1-Score de todas as classes, tratando cada classe com igual importância independente de sua frequência.
  - *F1-Score Micro*: Calculado globalmente contando o total de verdadeiros positivos, falsos negativos e falsos positivos, dando maior peso às classes mais frequentes.
  - *F1-Score Weighted*: Média ponderada do F1-Score de cada classe pelo número de amostras verdadeiras de cada classe.
- **Matriz de Confusão**: Uma tabela que visualiza o desempenho do modelo, mostrando as predições corretas e incorretas para cada classe, permitindo análise granular dos tipos de erro e identificação de classes com maior dificuldade de classificação.

O modelo é avaliado periodicamente durante o treinamento (a cada 10 épocas) em três conjuntos distintos: treinamento, validação e teste, proporcionando visão abrangente da evolução do desempenho e detecção precoce de *overfitting*.

5 Resultados Finais e Validação

5.1 Configuração Experimental

A configuração para a condução dos experimentos finais incluem (i) as métricas utilizadas para avaliação; (ii) os parâmetros para extração de conhecimento e representação em grafo; (iii) os parâmetros para o treinamento da GCN; (iv) os conjuntos de treinamento, validação e teste usados nos experimentos; (v) a configuração de infraestrutura de hardware e software na execução experimental.

Os resultados finais foram comparados com abordagens de classificação textual tradicionais e do estado da arte da literatura, incluindo os trabalhos relacionados mencionados no capítulo 2. Acurácia, F1-score (micro e macro), precisão e recall como métricas de comparação. O cálculo do F1-score micro considera todos os verdadeiros positivos, falsos positivos e falsos negativos de todos os documentos em conjunto, enquanto o F1-score macro é a média dos F1-scores calculados para cada classe de documento. Adicionalmente, geramos matrizes de confusão abrangentes e relatórios de classificação para fornecer uma análise de desempenho detalhada entre as diferentes categorias de documentos.

5.2 Abordagens para Extração de Conhecimento

O framework implementa duas metodologias distintas de extração de conhecimento para comparação: a primeira utiliza a api do GEMINI e a segunda emprega REBEL (Babelscape/rebel-large). A abordagem Gemini emprega um processo de extração em múltiplos passos: (1) Identificação de Entidades, usando princípios de reconhecimento de entidades nomeadas; (2) Identificação de Relações, através de padrões de análise de dependência; (3) Refinamento, ao resumir entidades e relações; e (4) Geração de Saída em formato de tripla padronizada: ('head', 'relation', 'tail').

A API Gemini foi configurada com uma temperatura de 0.7 para um equilíbrio entre criatividade e consistência, um máximo de 1024 tokens de saída, e implementou-se um recuo exponencial (exponential backoff) com até 3 tentativas de repetição por requisição. A limitação de taxa (rate limiting) é aplicada com um atraso de 1 segundo entre as chamadas da API para evitar o esgotamento da cota. Já a abordagem com REBEL processa o texto localmente, gerando triplas de conhecimento através de uma transformação de sequência para sequência e da análise especializada dos tokens de saída do modelo (marcadores <triplet>, <subj>, <obj>).

A configuração utilizada para o REBEL processa dados utilizando `max_length=216` e heurísticas de busca com `num_beams=3` e `num_return_sequences=3`. Salienta-se que esses valores não são os padrões de fábrica (defaults) da ferramenta na biblioteca *transformers*, mas foram deliberadamente definidos de forma empírica para balancear a qualidade de extração sem estourar o limite de decodificação sequencial durante processamento de textos pesados.

Ademais, durante o processamento de classificação, optou-se pela utilização de uma GCN padrão em oposição a uma *Relational GCN* (R-GCN). Reconhece-se que a GCN clássica trata as conexões genericamente, o que leva à perda de uma informação estrutural vital de um grafo heterogêneo: a tipagem relacional das arestas, ou seja, as diferenças de relacionamento implícitas das triplas. No entanto, o custo computacional atrelado a processar R-GCNs com vocabulários vastos demandaria alto *overhead* de paralelização matriz-tensor; assim, essa perda topológica foi aceita de modo a privilegiar um desempenho exequível nos mais de 100.000 documentos integrados.

5.3 Arquitetura da Rede Convolucional de Grafos

A arquitetura do modelo GCN é projetada para o processamento de grafos heterogêneos, incorporando documentos, palavras e entidades extraídas como tipos de nós distintos. A rede emprega *embeddings* de 300 dimensões para as representações iniciais dos nós, seguidas por duas camadas ocultas de 128 dimensões cada, com uma taxa de *dropout* de 0.5 para prevenir *overfitting*. Para o treinamento do modelo, utilizamos otimização Adam com uma taxa de aprendizado de 0.01, tamanho de lote (batch size) de 32, e um máximo de 300 épocas com uma paciência de *early stopping* de 20 épocas quando valor de *loss* não melhora.

O processo de construção do grafo integra arestas documento-palavra baseadas em TF-IDF com relacionamentos de entidades baseados em triplas de conhecimento. As características TF-IDF são limitadas a um vocabulário máximo de 5.000 termos para garantir eficiência computacional, mantendo a riqueza semântica. Os relacionamentos de entidades derivados da extração de conhecimento criam arestas adicionais no grafo, permitindo que a GCN aprenda simultaneamente a partir de informações lexicais e semânticas.

5.4 Configuração de Hardware e Software

Os experimentos são conduzidos em sistemas equipados com processadores Intel Core i7 (mínimo de 32 GB de RAM) e GPUs NVIDIA com suporte a CUDA para processamento acelerado de grafos. O ambiente de desenvolvimento principal utiliza Windows 10/11 com Python 3.8+, PyTorch 1.9+ e PyTorch Geometric para implementações de redes neurais de grafos.

O framework de software emprega a biblioteca transformers (versão 4.0+) para o gerenciamento do modelo REBEL, a biblioteca requests para comunicações com a API com mecanismos de fallback SSL, e o scikit-learn para o cálculo das métricas de avaliação.

Para experimentos em larga escala que excedem a capacidade computacional local, foi utilizado as máquinas disponíveis pelo Google Colab Pro + (adc mais infos aqui) e também foi implementado um processamento distribuído com gerenciamento automático de lotes e salvamento de *checkpoints*. As técnicas de otimização de memória incluem a construção dinâmica

de grafos, o carregamento seletivo de características e a alocação de recursos baseada em níveis para permitir o processamento de conjuntos de dados que variam de 100 a mais de 100.000 documentos.

5.5 Validação de Resultados

Nesta seção, apresentamos os resultados dos experimentos conduzidos nos quatro datasets selecionados: BBC-News (PT-Br), Reuters-21578, AG News e 20 Newsgroups. A análise foi estruturada para discutir três questões centrais: (1) o impacto do refinamento textual no desempenho do classificador; (2) a análise comparativa entre as abordagens de extração de conhecimento (Gemini vs. REBEL); e (3) o desempenho geral do framework *GenGraph* em comparação ao estado da arte (SOTA).

5.5.1 Impacto do Refinamento Textual

Para avaliar a eficácia da etapa de refinamento textual proposta via API Gemini, utilizamos o dataset Reuters-21578, reconhecido por seu desbalanceamento extremo de classes (na ordem de 3.926:1 entre a categoria mais frequente e as mais raras). Adicionalmente, 9 classes possuem menos de 2 amostras, o que dificulta a utilização de divisão estratificada durante a criação de conjuntos de treino, validação e teste.

O framework completo, incluindo a etapa de refinamento com Gemini, alcançou uma acurácia de teste de 81,78% e um F1-Macro de 54,43%. A análise por classe revelou desempenho excelente (F1 > 0,8) em classes frequentes como earn (91%), crude (87%) e acq (87%), mas resultados próximos a zero em classes com menos de 10 amostras, resultando em uma correlação de 0,67 entre a frequência da classe e o F1-score. O erro total de classificação foi de 18,22%, com as principais confusões ocorrendo entre categorias semanticamente relacionadas, como *acq* e *dlr* (notícias sobre aquisições e dólar, frequentemente sobrepostas em conteúdo financeiro).

Em comparação direta, o modelo executado sem a etapa de refinamento textual apresentou um desempenho marginalmente superior, com 81,85% de acurácia e um F1-Macro de 55,15%. Essa diferença, embora pequena, sugere que a etapa de refinamento do Gemini provou-se estatisticamente redundante para esta tarefa no dataset Reuters. A resultados indicam que o ganho de desempenho do framework é, na verdade, impulsionado por outros componentes, como a combinação de embeddings densos (BART) e arestas semânticas, que mitigam a necessidade de pré-refinamento em cenários de alta complexidade.

A decisão de restringir esta validação do refinamento textual exclusivamente ao dataset Reuters-21578 decorre de sua complexidade morfológica e do forte desbalanceamento de classe inerente às documentações de mercado financeiro ali contidas. Uma vez que o intuito era mensurar o efeito limpo das correções do Gemini na clareza gramatical e ortográfica das notícias sob forte ruído, restringimos a avaliação a ele para evitar altos custos logísticos de API. Como os

resultados apontaram o refinamento LLM como estatisticamente redundante no conjunto mais complexo testado, sua aplicação generalizada aos demais conjuntos foi mitigada.

5.5.2 Análise Comparativa de Extração de Conhecimento: Gemini vs. REBEL

A análise comparativa foca estritamente nas metodologias entre a API do Gemini e o REBEL. A escolha de delimitar os testes especificamente a essas duas abordagens embasa-se na necessidade de confrontar dois paradigmas arquiteturais polarizados que definem os cenários atuais de extração: de um lado, a generalização escalável baseada em instrução via prompt processada em nuvem com alta volumetria de parâmetros (Gemini); do outro, a precisão *end-to-end* de um modelo menor, específico para sequenciamento sintático local (REBEL). Os conjuntos Reuters, AG News e 20 Newsgroups foram utilizados como base para este experimento.

A performance do dataset AG News apresentou um dos melhores resultados durante o treinamento. Devido ao custo computacional e a limitação de quotas/tokens por minuto na api do Gemini, o experimento limitou-se a 10.000 documentos. Os resultados na Figura 21 apresentam os resultados de *loss* (Função de perda) e acurácia do decorrer das épocas na etapa de treinamento do modelo utilizando a abordagem de REBEL + GCN. Nota-se que os resultados de acurácia são relativamente estavéis (90.35% e 87,55% para treino e validação, respectivamente (Tabela 3), com pequenas flutuações ao longo das épocas subsequentes, enquanto a perda de treinamento continua a diminuir, isso se dá pois o dataset possui classes balanceadas. E também sugere que o modelo converge rapidamente e evita *overfitting* significativo nas épocas iniciais. A similaridade entre F1-macro, micro e weighted corrobora a natureza balanceada do dataset AG News, indicando que o modelo não tem um viés forte em relação a nenhuma classe específica.

Observa-se que as classes 'Sports' e 'World' apresentam os melhores resultados (Figura 22), com altas taxas de acerto na matriz de confusão (484 para Sports, 436 para World). A matriz de confusão também indica que a maior parte dos erros ocorre entre as classes 'Business' e 'Sci/Tech', com 71 instâncias de 'Business' classificadas incorretamente como 'Sci/Tech' e 40 instâncias de 'Sci/Tech' classificadas como 'Business'. Isso sugere que pode haver alguma sobreposição conceitual ou semântica entre essas duas categorias que o modelo tem mais dificuldade em distinguir, em comparação com as outras classes.

Figura 21 – Gráfico linear do valor de custo e acurácia durante o experimento - AG News

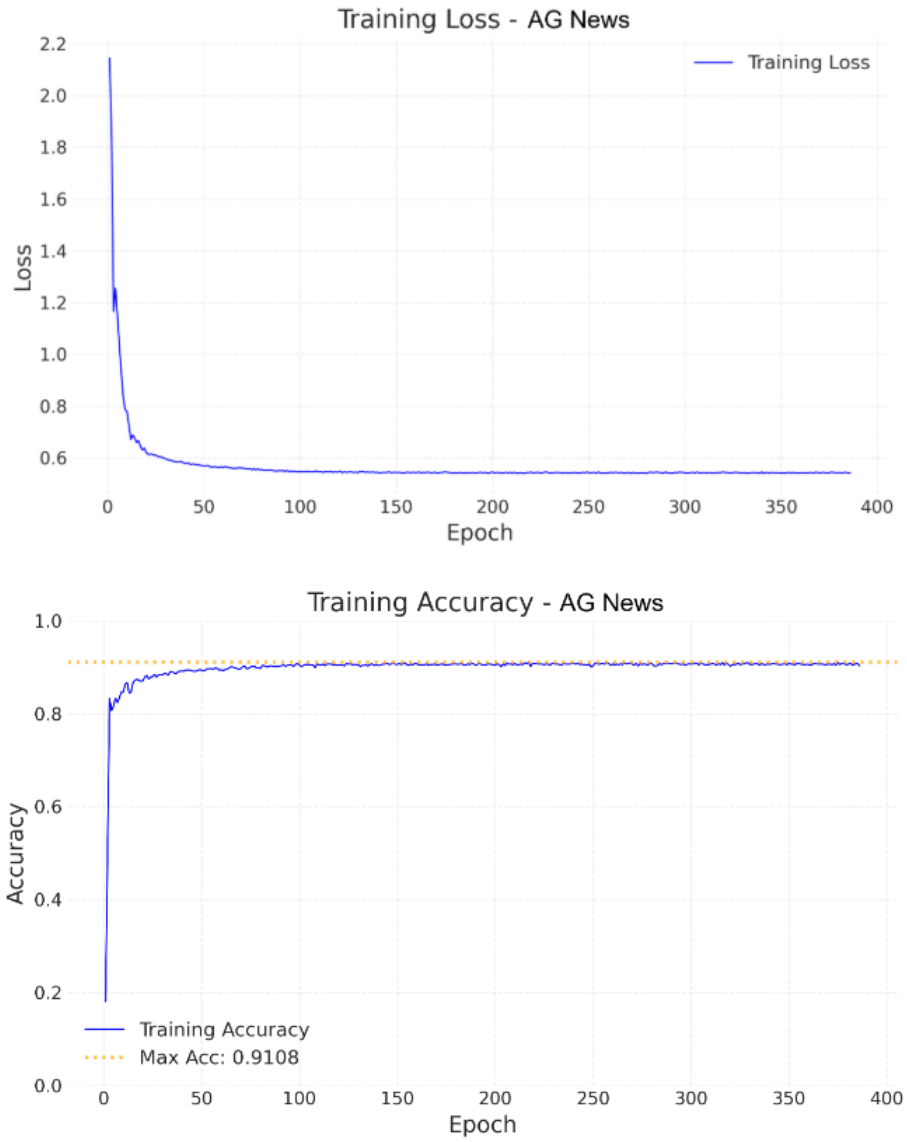


Figura 22 – Matriz de confusão - AG News

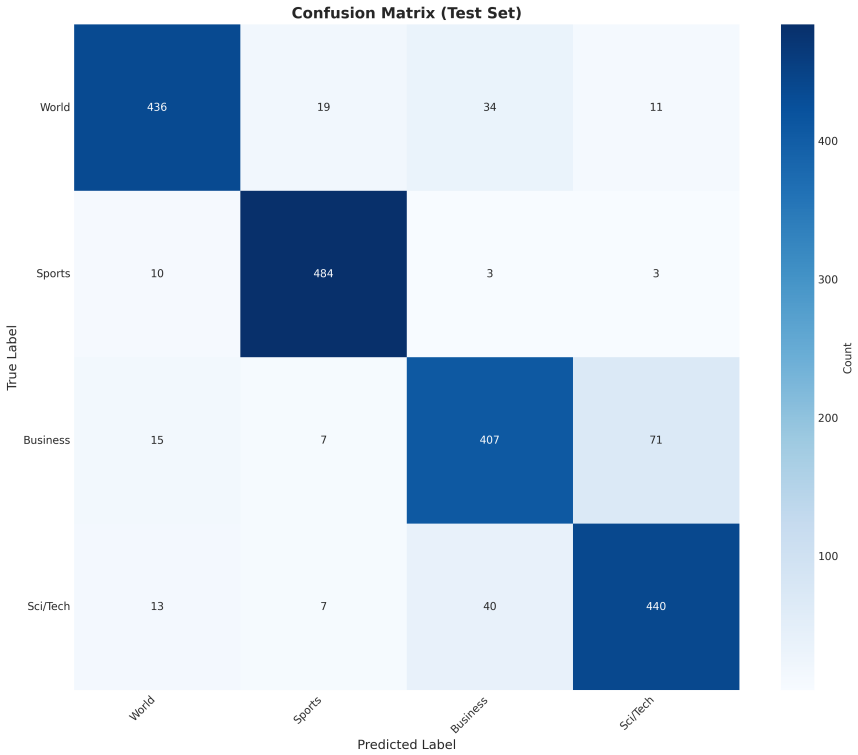


Tabela 3 – Resultados do Treinamento - Rebel + GCN

Loss		**Accuracy**		**F1-Score**	
Train	**Val.**	**Train**	**Val.**	**Train**	**Val.**
0.5435	0.6096	0.9035	0.8755	0.9034	0.8753

Fonte: Elaborada pelo autor.

Para a abordagem utilizando Gemini como extrator de conhecimento o modelo alcançou 0.8351 de acurácia nos subconjuntos de validação e 0.9980 durante o treinamento em sua melhor época. Os valores de *loss* se mostraram estaveis em uma constante queda ao longo da épocas, aqui quanto menor o valor é melhor. A figura 23 exibe o comportamento descrito do modelo durante o treinamento e validação.

Figura 23 – Gráfico linear do valor de custo e acurácia durante o experimento - Gemini + GCN - AG News

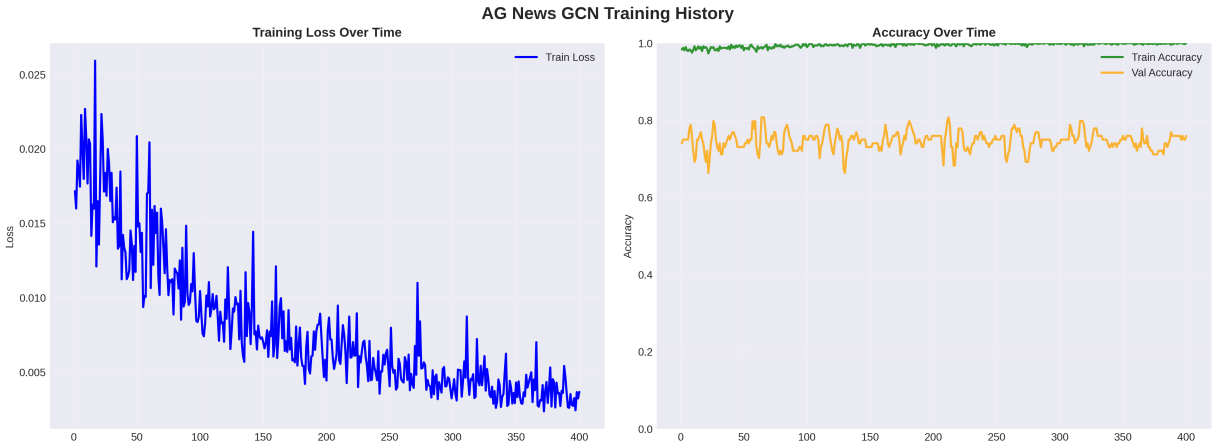


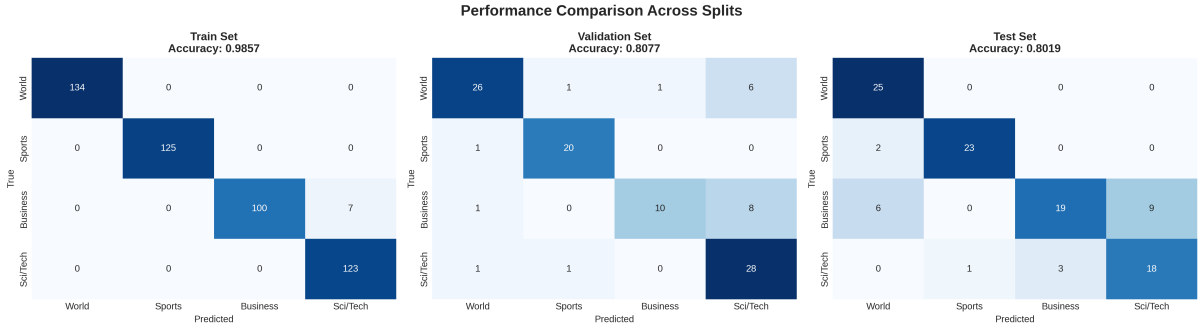
Tabela 4 – Resultados do Treinamento - Gemini + GCN - AG News

Split	Accuracy	Precision	Recall	F1-Score
Train	0.9857	0.9865	0.9857	0.9856
Validation	0.8077	0.8351	0.8077	0.8038
Test	0.8019	0.8201	0.8019	0.7949

Fonte: Elaborada pelo autor.

A Figura 24 exibe as matrizes de confusão para os conjuntos de treino, teste e validação, e as respectivas acurácias da melhor época do modelo. Verifica-se novamente que as classes 'Sports' e 'World' obtiveram os melhores desempenhos em todos os conjuntos. Em contrapartida, as classes 'Business' e 'Sci/Tech' apresentaram quedas significativas de performance, indicando que o modelo falhou em identificar quase metade das instâncias reais desta categoria.

Figura 24 – Matriz de confusão - Gemini + GCN - AG News



Os resultados indicam uma performance parecida entre as duas abordagens, com uma leve vantagem para o REBEL em certos cenários, mas sem diferenças extremas. A abordagem com REBEL demonstrou maior estabilidade e melhor generalização, alcançando uma acurácia

de validação de 87,65%, superando os 83,51% obtidos pela abordagem que utiliza a API do Gemini.

Já para o dataset Reuters, a análise comparativa indica uma equivalência técnica substancial entre as abordagens de extração de conhecimento. Conforme detalhado nas Tabelas 5 e 6, a diferença de desempenho no conjunto de teste é marginal, com a abordagem REBEL alcançando uma acurácia ligeiramente superior de 0.8185 em comparação aos 0.8165 obtidos pelo Gemini. Em termos de robustez para classes desbalanceadas, observa-se um empate técnico virtual no F1-Macro, registrando 0.5515 para o REBEL e 0.5517 para o Gemini. Embora o REBEL tenha demonstrado um ajuste marginalmente mais forte aos dados de treinamento (0.8960 vs 0.8888), ambos os modelos convergiram para uma performance de validação praticamente idêntica (aproximadamente 0.834). Estes resultados sugerem que, para a complexidade semântica do Reuters, a aplicação de refinamento textual e extração via LLM generativo não traduz o custo computacional adicional em ganhos estatísticos de performance quando comparado à extração direta do REBEL.

Tabela 5 – Resultados do Treinamento - Rebel + GCN - Reuters

Metric	Training (Final Epoch)	Validation (Best Epoch)	Test
Loss	0.8039	0.9989	1.0174
Accuracy	0.8960	0.8343	0.8185
F1-Macro	0.7920	0.5890	0.5515

Fonte: Elaborada pelo autor.

Tabela 6 – Resultados do Treinamento - Gemini + GCN - Reuters

Metric	Training (Final Epoch)	Validation (Best Epoch)	Test
Loss	0.8128	0.9977	1.0144
Accuracy	0.8888	0.8348	0.8165
F1-Macro	0.8060	0.6189	0.5517

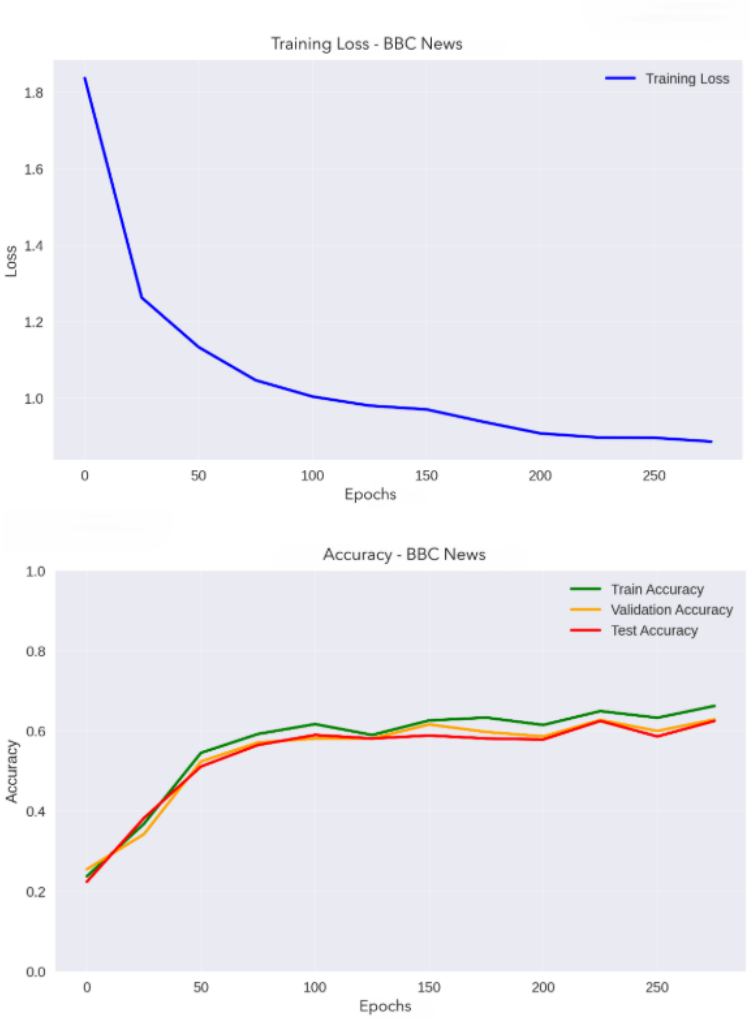
Fonte: Elaborada pelo autor.

5.5.3 Desempenho Geral e Comparação com o Estado da Arte

Por fim, avaliamos a capacidade de generalização do framework *GenGraph* e sua posição em relação a baselines da literatura, analisando os datasets BBC News e 20 Newsgroups.

Para o dataset BBC-News, o modelo apresentou um processo de treinamento estável, com a curva de perda declinando de 1,85 para 0,9 (Figura 25). Embora as curvas de treino e validação indiquem ausência de *overfitting*, o desempenho foi considerado abaixo do esperado, sugerindo que o modelo atingiu sua capacidade máxima precocemente neste dataset específico. A acurácia, que começou com valor de aproximadamente 0.25, melhorou rapidamente para 0.5 nas primeiras 50 épocas, porém estabilizou-se em um valor final de aproximadamente 0.6250.

Figura 25 – Gráfico linear do valor de custo e acurácia durante o experimento - BBC News



O desempenho do modelo no 20 Newsgroups apresentou o melhor desempenho relativo do estudo. Na Figura 26, observa-se uma redução drástica na perda de validação, que partiu de um valor inicial alto de aproximadamente 4.0699 para 1.2075 já na décima época, convergindo para uma estabilidade em torno de 0.8053 ao final do processo. De forma correspondente, a acurácia de validação demonstrou um crescimento acelerado, iniciando em 0.3691 e alcançando seu pico de 0.9695 na época 226 (melhor época) e 0.9691 na última época (Tabela 7). Os valores de acurácia de validação sendo superiores aos de treinamento indicam uma generalização robusta e a eficácia das técnicas de regularização aplicadas. Neste dataset, o GenGraph superou baselines como TextGCN (86,34%) e ChatGraph (79,15%) (Tabela 8), validando a robustez da arquitetura proposta.

Tabela 7 – Desempenho geral do modelo nos subconjuntos de Treino, Validação e Teste - 20 Newsgroups

Loss		Accuracy		F1-Score	
Train	**Val.**	**Train**	**Val.**	**Train**	**Val.**
0.8683	0.8053	0.9382	0.9691	0.9352	0.9673

Fonte: Elaborada pelo autor.

Figura 26 – Valores de *loss* e acurácia ao longo de épocas

Analisando o panorama de métricas obtidas, conclui-se que o framework *GenGraph* supera de forma inequívoca o estado da arte consolidado exclusivamente no tratamento metodológico de tópicos complexos como o *corpus* do *dataset* 20 Newsgroups. Neste subconjunto, a capacidade da arquitetura em agregar o sequenciamento semântico do REBEL ao fluxo convolucional foi crucial: o modelo proposto atingiu excepcionais 96,91% de acurácia, suplantando os 86,34% registrados pelo pioneiro TextGCN e os 79,15% estabelecidos pelo modelo paralelo ChatGraph (Tabela 8).

Tabela 8 – Comparação de diferentes técnicas e *frameworks*

Technique/Framework	**BBC News**	**20NG**	**Reuters**	**AG News**
OPHELIA FFNN	0,985	-	0,869	0,919
OPHELIA CapsNet	0,978	-	0,778	0,91
TextGCN (2 layers)	-	0,8634	0,9356	-
ChatGraph (with TF-IDF)	-	0,7915	0,9214	-
TF-IDF+LR	-	0,8319	NaN	0,8695
GenGraph (REBEL+GCN)	0,6250	**0,9691**	0,8185	0,8765*
GenGraph (Gemini+GCN)	-	N/A*	0,8348	**0,8351**

Fonte: Elaborada pelo autor.

*Os resultados da abordagem GenGraph (Gemini) não foram coletados devido a limitação de quotas tokens da API Gemini

6 Considerações Finais

O estudo explorou a interseção entre o Processamento de Linguagem Natural (PLN) e a Aprendizagem de Máquina em Grafos, explorando como a estruturação do conhecimento humano pode potencializar arquiteturas neurais. Partindo da premissa de que a representação semântica enriquecida é fundamental para a compreensão textual, este estudo propôs o framework *GenGraph*, uma abordagem híbrida que integra a capacidade generativa de Large Language Models (LLMs), especificamente o Gemini, e extratores especializados como o REBEL, com a robustez topológica das Graph Convolutional Networks (GCNs). Ao longo desta trajetória, buscou-se não apenas implementar uma solução técnica, mas compreender as dinâmicas de interação entre modelos generativos e discriminativos em cenários de classificação.

No que tange os objetivos estabelecidos, considera-se que o objetivo geral foi atingido com o desenvolvimento e validação do framework proposto. A pesquisa respondeu ao problema central sobre a viabilidade e eficácia da integração GNN-LLM, demonstrando que a inserção de conhecimento estruturado (triplas de entidades e relações) em grafos heterogêneos pode, de fato, elevar métricas de desempenho em contextos de alta dimensionalidade, como evidenciado nos resultados superlativos do dataset 20 Newsgroups (96,91% de acurácia). Por outro lado, o estudo revelou que a eficácia não é constante, variando significativamente conforme a natureza do *dataset* e a complexidade semântica das classes, o que amplia a compreensão do problema para além da simples "aplicação de tecnologia", inserindo-o no campo da "adequação topológica".

Para além dos resultados práticos, este trabalho entrega uma contribuição importante para comunidade de pesquisa científica: a disponibilização de um dataset inédito e público de grafos de conhecimento. Este repositório consolida as extrações realizadas pelas abordagens REBEL e Gemini para os benchmarks Reuters-21578, BBC News e AG News. Ao fornecer este recurso, o estudo mitiga o alto custo computacional inicial de extração para futuros pesquisadores, oferecendo uma base sólida para a validação de novas hipóteses e o *benchmarking* de arquiteturas baseadas em grafos.

A verificação das hipóteses conduziu a achados científicos relevantes, tanto pela confirmação quanto pela refutação. A hipótese de que o refinamento textual via modelos generativos (Gemini) resultaria em métricas superiores de classificação foi refutada no contexto do dataset Reuters-21578, onde o processo se mostrou estatisticamente redundante frente à robustez dos *embeddings* densos (BERT). Longe de ser um insucesso, este dado é uma descoberta valiosa, sugerindo que a complexidade computacional do pré-processamento generativo nem sempre se traduz em ganho de performance. Paralelamente, a hipótese sobre a competência dos extratores foi refinada: embora o Gemini seja capaz de extrair conhecimento, a abordagem especializada do REBEL demonstrou maior estabilidade e eficiência, confirmando que modelos *end-to-end* específicos ainda possuem vantagens sobre modelos generalistas em tarefas estruturadas.

Do ponto de vista metodológico e teórico, a abordagem de grafos heterogêneos mostrou-

se robusta, amparada por um referencial teórico que dialoga com o estado da arte, como o TextGCN e ChatGraph. Contudo, é imperativo exercer a autocrítica científica: a metodologia enfrentou limitações impostas por restrições de infraestrutura e cotas de API, o que restringiu a escala dos experimentos no AG News e impediu a coleta completa de dados do Gemini para todos os datasets. Ademais, o desempenho moderado no BBC News (estabilização em 62,50%) sinaliza uma limitação na capacidade do modelo proposto em generalizar para datasets com características específicas de distribuição, tamanho e idioma, sugerindo que a arquitetura atual pode atingir um platô de aprendizado precocemente em certos cenários.

Em suma, esta dissertação demonstrou que a integração de *Large Language Models* e *Graph Neural Networks* oferece uma via promissora para otimizar a classificação de texto e a extração de conhecimento. O framework *GenGraph* cumpriu o objetivo de preencher a lacuna existente na modelação de grafos heterogêneos, provando que a injeção de conexões semânticas explícitas — extraídas via arquiteturas *seq2seq* estruturadas — enriquece a topologia do *corpus* sem incorrer em custos computacionais proibitivos. Ao superar o estado da arte em cenários de alta complexidade lexical, o modelo reafirma que o futuro do Processamento de Linguagem Natural não reside apenas na escala dos parâmetros, mas na integração inteligente entre o raciocínio factual dos LLMs e o poder de propagação estrutural das GNNs. O conjunto de dados e as metodologias aqui consolidadas abrem um novo precedente analítico, fornecendo à comunidade científica uma base robusta para futuras investigações em sistemas de inteligência artificial. Para a evolução do *GenGraph*, melhorias futuras incluem: (i) a exploração de arquiteturas de grafos mais dinâmicas, como *Graph Attention Networks* (GATs); (ii) a investigação qualitativa dos casos de falha no refinamento textual; (iii) expansão dos experimentos para incluir outros LLMs de código aberto, mitigando as limitações de API proprietárias; e (iv) a avaliação de métodos alternativos de vetorização que possam substituir ou complementar os *embeddings* do BERT, buscando maior eficiência computacional.

Referências

AGARWAL, M. An overview of natural language processing. *International Journal for Research in Applied Science and Engineering Technology*, v. 7, n. 5, p. 2811–2813, may 2019. Citado na página 12.

ANIL, R. et al. Gemini: A family of highly capable multimodal models. 12 2023. Citado na página 12.

BAO, Y.; XU, W.; WANG, L. Design and implementation on citation network link prediction system based on gat. In: ____. [S.l.]: IOS Press Ebooks, 2024. ISBN 9781643684802. Citado na página 35.

CABOT, P.-L. H.; NAVIGLI, R. REBEL: Relation extraction by end-to-end language generation. In: MOENS, M.-F. et al. (Ed.). *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021. p. 2370–2381. Disponível em: <https://aclanthology.org/2021.findings-emnlp.204>. Citado 3 vezes nas páginas 22, 36 e 44.

CORONA, N.; HERNáNDEZ-COLíN, D.; BUSTILLO-HERNáNDEZ, C. Model of natural semantic space for ontologies' construction. *IJCOPI*, v. 3, p. 93–108, 05 2012. Citado 2 vezes nas páginas 19 e 20.

COSTA, L. S. d. *OPHELIA - A Neural Solution for Text Classification using Joint Embeddings of Words and KG Entities*. Tese (Tese de Doutorado) — Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brasil, 2023. Orientador: Prof. Renato Fileto. Citado na página 15.

COSTABELLO, L. et al. *AmpliGraph: a Library for Representation Learning on Knowledge Graphs*. 2019. Disponível em: <https://doi.org/10.5281/zenodo.2595043>. Citado na página 26.

DONG, S.; WANG, P.; ABBAS, K. A survey on deep learning and its applications. *Computer Science Review*, v. 40, p. 100379, 2021. ISSN 1574-0137. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574013721000198>. Citado na página 13.

GAO, H. et al. Triple context-based knowledge graph embedding. *IEEE Access*, v. 6, p. 58978–58989, 2018. Citado na página 21.

GAUTAM, A.; KUMAR, A.; SHAH, R. R. Crises multi modal disaster. In: *Proceedings of the IEEE International Conference on Multimedia Big Data (BigMM)*. Singapore: IEEE, 2019. Citado na página 40.

HOGAN, A. et al. *Knowledge Graphs*. Springer, 2021. (Synthesis Lectures on Data, Semantics, and Knowledge, 22). ISBN 9783031007903. Disponível em: <https://kgbook.org/>. Citado na página 20.

HU, W. et al. *Open Graph Benchmark: Datasets for Machine Learning on Graphs*. 2021. Disponível em: <https://arxiv.org/abs/2005.00687>. Citado na página 27.

JI, S. et al. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, PP, 04 2021. Citado na página 19.

JI, S. et al. A survey on knowledge graphs: representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, Institute of Electrical and Electronics Engineers, v. 33, n. 2, p. 494–514, fev. 2022. ISSN 2162-237X. Publisher Copyright: IEEE Copyright: Copyright 2021 Elsevier B.V., All rights reserved. Citado 2 vezes nas páginas 23 e 24.

JIN, Z. Principle, methodology and application for data cleaning techniques. *BCP Business Management*, v. 26, p. 724–732, 09 2022. Citado na página 18.

KOWSARI, K. et al. Text classification algorithms: A survey. *Information (Switzerland)*, v. 10, 04 2019. Citado 2 vezes nas páginas 36 e 39.

LIN, T.-Y. et al. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, p. 1–1, 07 2018. Citado na página 50.

LIN, Y. et al. Knowledge representation learning: A quantitative review. *ArXiv*, abs/1812.10901, 2018. Disponível em: <https://api.semanticscholar.org/CorpusID:57189248>. Citado na página 25.

MINAEE, S. et al. Large language models: A survey. *ArXiv*, abs/2402.06196, 2024. Disponível em: <https://api.semanticscholar.org/CorpusID:267617032>. Citado 3 vezes nas páginas 18, 19 e 36.

NARASIMHAN, M.; LAZEBNIK, S.; SCHWING, A. Out of the box: Reasoning with graph convolution nets for factual visual question answering. *Advances in Neural Information Processing Systems*, v. 2018-December, p. 2654–2665, 2018. ISSN 1049-5258. Publisher Copyright: © 2018 Curran Associates Inc.All rights reserved.; 32nd Conference on Neural Information Processing Systems, NeurIPS 2018 ; Conference date: 02-12-2018 Through 08-12-2018. Citado na página 33.

NAVEED, H. et al. A comprehensive overview of large language models. Association for Computing Machinery, New York, NY, USA, v. 16, n. 5, ago. 2025. ISSN 2157-6904. Disponível em: <https://doi.org/10.1145/3744746>. Citado na página 12.

S., A.; KRISHNA, C.; GOVINDARAJAN, U. H. Graph embedding approaches for social media sentiment analysis with model explanation. *International Journal of Information Management Data Insights*, v. 4, p. 100221, 04 2024. Citado na página 15.

SHI, Y. et al. *ChatGraph: Interpretable Text Classification by Converting ChatGPT Knowledge to Graphs*. 2023. Disponível em: <https://arxiv.org/abs/2305.03513>. Citado 2 vezes nas páginas 15 e 42.

SUN, C. et al. Attention-based graph neural networks: a survey. *Artificial Intelligence Review*, v. 56, n. 2, p. 2263–2310, November 2023. ISSN 1573-7462. Disponível em: <https://doi.org/10.1007/s10462-023-10577-2>. Citado na página 31.

SUN, K. et al. Relational structure-aware knowledge graph representation in complex space. *Mathematics*, v. 10, n. 11, 2022. ISSN 2227-7390. Disponível em: <https://www.mdpi.com/2227-7390/10/11/1930>. Citado na página 22.

VELIčKOVIć, P. et al. Graph attention networks. In: *International Conference on Learning Representations*. [s.n.], 2018. Disponível em: <https://openreview.net/forum?id=rJXMpikCZ>. Citado na página 34.

WANG, K.; DING, Y.; HAN, S. C. Graph neural networks for text classification: a survey. *Artificial Intelligence Review*, v. 57, n. 8, p. 190, jul. 2024. ISSN 1573-7462. Disponível em: <https://doi.org/10.1007/s10462-024-10808-0>. Citado na página 37.

WANG, Q. et al. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, v. 29, n. 12, p. 2724–2743, 2017. Citado 2 vezes nas páginas 20 e 21.

WANG, Y. et al. Text fcg: Fusing contextual information via graph learning for text classification. *Expert Systems with Applications*, v. 219, p. 119658, 2023. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417423001598>. Citado na página 16.

WU, L. et al. Graph neural networks. In: ____. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Nature Singapore, 2022. p. 27–37. ISBN 978-981-16-6054-2. Disponível em: <https://doi.org/10.1007/978-981-16-6054-2_3>. Citado 6 vezes nas páginas 22, 23, 25, 26, 27 e 28.

WU, Z. et al. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, Institute of Electrical and Electronics Engineers, v. 32, n. 1, p. 4–24, jan. 2021. ISSN 2162-237X. Citado 3 vezes nas páginas 35, 37 e 38.

YAO, L.; MAO, C.; LUO, Y. Graph convolutional networks for text classification. In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2019. (AAAI'19/IAAI'19/EAAI'19). ISBN 978-1-57735-809-1. Disponível em: <https://doi.org/10.1609/aaai.v33i01.33017370>. Citado 2 vezes nas páginas 33 e 36.

YE, Z. et al. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, v. 10, p. 1–1, 01 2022. Citado 2 vezes nas páginas 12 e 28.

ZAIB, M. et al. Conversational question answering: a survey. *Knowledge and Information Systems*, v. 64, n. 12, p. 3151–3195, December 2022. ISSN 0219-3116. Disponível em: <https://doi.org/10.1007/s10115-022-01744-y>. Citado na página 24.

ZHANG, S. et al. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, v. 6, 11 2019. Citado 2 vezes nas páginas 32 e 33.

ZHOU, J. et al. Graph neural networks: A review of methods and applications. *AI Open*, v. 1, p. 57–81, 2020. ISSN 2666-6510. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666651021000012>. Citado 9 vezes nas páginas 21, 25, 28, 29, 30, 31, 32, 34 e 38.

ZHU, J. et al. A semi-supervised model for knowledge graph embedding. *Data Mining and Knowledge Discovery*, v. 34, 01 2020. Citado na página 20.

DADOS CURRICULARES

IDENTIFICAÇÃO	
	Larissa Pessoa Sá Menezes 20/02/1999
Nacionalidade	brasileira
Nome em citações bibliográficas:	Menezes, Larissa Pessoa Sá Menezes, L. P. S.
Currículo Lattes	<http://lattes.cnpq.br/4277029770540460>
ORCID	<https://orcid.org/0000-0001-5661-1669>
FORMAÇÃO ACADÊMICA	
2016/2021	Graduação em Análise e Desenvolvimento de Sistemas Instituto Federal de Educação, Ciência e Tecnologia do Amazonas – IFAM
2023/2026	Mestrado em Ciência da Computação Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP)
PRODUÇÃO BIBLIOGRÁFICA	
MENEZES, L. P. S.; BREVE, F. Integração de Large Language Models e Graph Neural Networks para Otimizar Classificação de Texto e Extração de Conhecimento. In: XIII Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP (WPPGCC 2024), 26 e 27 de Novembro de 2024.	
MENEZES, L. P. S.; ORLANDINI, M. E.; BREVE, F. Algoritmo do vagalume e algoritmo híbrido: otimização de funções de referência através da aplicação de lógicas bioinspiradas. In: XII Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP (WPPGCC 2023), 21 e 22 de Novembro de 2023.	
PARTICIPAÇÃO EM EVENTOS CIENTÍFICOS	
XIII Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP (WPPGCC 2024). Apresentação de trabalho. 26 e 27 de Novembro de 2024.	
XII Workshop do Programa de Pós-Graduação em Ciência da Computação da UNESP (WPPGCC 2023). Apresentação de trabalho. 21 e 22 de Novembro de 2023.	
V Mostra Interdisciplinar de Extensão do IFAM/Campus Manaus Centro (V MIEX/IFAM/CMC). Proposta de Glossário Visual de LIBRAS para Disciplina de Banco de Dados. 2018. (Exposição).	
Women Techmakers Manaus 2018. 2018. (Congresso).	