UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

Alexandre dos Santos Gualberto

Predição de Links em Grafos Bipartidos para Recomendação de Empregos

São Carlos
2025

Alexandre dos Santos Gualberto

Predição de Links em Grafos Bipartidos para Recomendação de Empregos

Trabalho de Conclusão de Curso de Graduação em Engenharia de Computação do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de São Carlos como requisito para a obtenção do título de Engenheiro de Computação.
Orientador: Prof. Dr. Murilo Coelho Naldi
Coorientador: Prof. Dr. Alan Demétrius Baria Valejo

São Carlos
2025

Gualberto, Alexandre dos Santos

Predição de Links em Grafos Bipartidos para Recomendação de Empregos / Alexandre dos Santos Gualberto -- 2025.
35f.

TCC (Graduação) - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador (a): Murilo Coelho Naldi
Banca Examinadora: Alan Demétrius Baria Valejo, Heloisa Camargo
Bibliografia

1. Sistemas de Recomendação. 2. Redes Neurais de Grafos. 3. Modelos de Linguagem de Larga Escala. I. Gualberto, Alexandre dos Santos. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Arildo Martins - CRB/8 7180

Alexandre dos Santos Gualberto

Predição de Links em Grafos Bipartidos para Recomendação de Empregos

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Computação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Computação.

São Carlos, 20 de Fevereiro de 2025.

Banca Examinadora:

Prof. Dr. Murilo Coelho Naldi
Universidade Federal de São Carlos

Prof. Dr. Alan Demétrius Baria Valejo
Universidade Federal de São Carlos

Prof. Dra. Heloisa Camargo
Universidade Federal de São Carlos

RESUMO

O presente trabalho desenvolve e avalia um sistema de recomendação de oportunidades profissionais que integra informações textuais e estruturais, utilizando modelos de linguagem de grande escala e redes neurais de grafos. O estudo teve como objetivo principal criar um framework preditivo capaz de superar as limitações de abordagens tradicionais, combinando as representações semânticas obtidas por meio de embeddings de texto com a capacidade das redes neurais de grafos de explorar padrões de interação em grafos bipartidos. Para tanto, o projeto implementou três abordagens distintas: uma baseada exclusivamente em embeddings gerados por modelos de linguagem, outra fundamentada apenas na estrutura do grafo, e uma terceira que unifica as duas metodologias. Durante os experimentos, o sistema foi treinado para realizar a predição de arestas que representam candidaturas, utilizando técnicas de amostragem negativa, mini-batch training e estratégias de otimização que garantiram a eficiência do processo de aprendizado. Os resultados demonstraram que a abordagem híbrida alcançou desempenho superior em métricas como AUC, acurácia, precisão, revocação e F1-Score, evidenciando a importância da integração entre o contexto semântico e a estrutura relacional dos dados. O estudo também destaca que, embora o método híbrido apresente desafios relacionados à disponibilidade de dados de qualidade e à demanda computacional, o investimento nesses aspectos é compensado pela melhoria na precisão das recomendações. Conclui-se que a combinação de técnicas baseadas em linguagem natural e aprendizado em grafos constitui uma estratégia promissora para sistemas de recomendação em ambientes complexos, contribuindo para uma melhor adequação entre candidatos e vagas e para a evolução dos processos de recrutamento e seleção.

Palavras-chave: Recomendação de vagas; Modelos de Linguagem; Redes Neurais de Grafos.

ABSTRACT

This work develops and evaluates a recommendation system for job opportunities that integrates both textual and structural information, employing large-scale language models and graph neural networks. The main objective is to create a predictive framework that overcomes the limitations of traditional approaches by combining semantic representations obtained from textual embeddings with the ability of graph neural networks to capture interaction patterns within bipartite graphs. To achieve this, three distinct approaches were implemented: one based solely on text embeddings generated by language models, another founded exclusively on the graph structure, and a third that unifies both methodologies. The system was trained to predict links representing job applications using negative sampling, mini-batch training, and optimization strategies that ensured an efficient learning process. The results indicated that the hybrid approach achieved superior performance in terms of AUC, accuracy, precision, recall, and F1-Score, underscoring the importance of integrating semantic context with relational structure. Although the hybrid method presents challenges related to data quality and computational demands, these investments are justified by the improvements in recommendation accuracy. The study concludes that combining natural language processing techniques with graph-based learning constitutes a promising strategy for recommendation systems in complex environments, contributing to a better match between candidates and job vacancies and advancing recruitment processes.

Keywords: Job recommendation; Language Models; Graph Neural Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	Área sob a curva ROC
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GenAI	Inteligência Artificial Generativa
GNN	Redes Neurais de Grafos
GraphSAGE	Graph Sample and Aggregate
LLM	Modelos de Linguagem de Grande Escala
MLP	Perceptron multicamadas

SUMÁRIO

1 INTRODUÇÃO

Com a crescente popularidade do recrutamento online, plataformas como LinkedIn e BeeCrowd publicam diariamente um grande volume de oportunidades de emprego, o que torna desafiador filtrar e conectar candidatos às vagas mais adequadas. Em resposta a essa demanda, os sistemas de recomendação têm se consolidado como ferramentas fundamentais para lidar com a sobrecarga de informações, auxiliando diferentes plataformas a fornecer conteúdo relevante e personalizado aos usuários (JANNACH; ZANKER, 2022; WU, S. *et al.*, 2022; RESNICK; VARIAN, 1997; WU, Z. *et al.*, 2021). Tradicionalmente, no contexto da busca por oportunidades de trabalho, tais sistemas são modelados como tarefas de previsão de arestas em grafos bipartidos, nos quais os nós representam candidatos e vagas, enquanto as arestas indicam interações, como candidaturas ou correspondências de perfis (BERG; KIPF; WELLING, 2017).

Abordagens clássicas, como filtragem colaborativa (SARWAR *et al.*, 2001; XIE *et al.*, 2021) e fatoração de matrizes (KOREN; BELL; VOLINSKY, 2009; BENNETT; LANNING, 2007), ainda que amplamente difundidas, enfrentam restrições em cenários de escassez de dados, alta dimensionalidade ou interações complexas — por exemplo, quando é necessário considerar histórico profissional, competências técnicas e afinidades culturais entre candidato e empresa (KABBUR; NING; KARYPIS, 2013). Em vista desses desafios, pesquisas recentes têm investigado diferentes estratégias. Entre elas, estão propostas que utilizam GNN (Redes Neurais de Grafos) para aperfeiçoar a correspondência entre usuários e vagas (LIU *et al.*, 2024), explorando a topologia inerente ao grafo de interações candidato–vaga. De forma complementar, surgiram sistemas inteiramente baseados em GenAI (Inteligência Artificial Generativa), como o de (ZHENG *et al.*, 2023), no qual LLM (Modelos de Linguagem de Grande Escala) são empregados para atribuir um *score* de aderência entre o currículo e a descrição de uma vaga.

Para superar as limitações de cada abordagem quando tomadas isoladamente, este projeto propõe a implementação de um sistema de recomendação de empregos que conjuga *embeddings* provenientes de LLM com algoritmos de GNN (WANG *et al.*, 2019). De um lado, LLM permitem extrair, de modo automático, representações de texto que encapsulam características latentes de descrições de vagas e de perfis profissionais, ampliando a análise para além de dados numéricos. De outro, GNN aproveitam a estrutura do grafo bipartido para capturar interações complexas, viabilizando previsões de arestas mais acuradas e robustas mesmo em cenários de dados escassos. Dessa forma, alia-se o poder de entendimento contextual proporcionado por LLM — recentemente popularizado com o advento de sistemas como o ChatGPT — à capacidade das GNN de refletir as relações estruturais em grande escala.

Com essa combinação de tecnologias, espera-se desenvolver uma ferramenta de recomendação avançada que ofereça resultados mais precisos e pertinentes, contribuindo ao mesmo tempo para o progresso do estado da arte em sistemas de recomendação e para a evolução digital do mercado de trabalho.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo geral deste projeto é desenvolver um modelo preditivo robusto para a recomendação de oportunidades profissionais. Para tanto, propõe-se realizar uma análise aprofundada do comportamento histórico dos candidatos e das características contextuais de cada vaga e usuário, integrando, de forma sinérgica, técnicas avançadas de LLM e GNN. Ao final, pretende-se disponibilizar um *framework* inovador que combine a extração automática de conhecimento textual com a modelagem precisa de relacionamentos, resultando em recomendações mais eficazes, personalizadas e alinhadas às demandas do mercado de trabalho.

1.1.2 Objetivos Específicos

Para atingir o objetivo geral, o projeto propõe:

- Representar as relações entre candidatos, vagas e aplicações por meio de um *grafo bipartido* (*candidate–job*), possibilitando a identificação e análise de padrões de interação relevantes;
- Empregar LLM para extrair, de maneira automatizada, características semânticas de descrições de vagas e perfis de candidatos, enriquecendo o modelo com informações contextuais valiosas;
- Utilizar GNN para capturar as interações estruturais presentes no grafo, aprimorando a tarefa de *link prediction* e, consequentemente, a qualidade das recomendações;
- Validar o desempenho do modelo preditivo por meio de métricas apropriadas, comparando-o com abordagens diferentes, a fim de demonstrar a eficácia do framework proposto.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMAS DE RECOMENDAÇÃO

Sistemas de recomendação têm como objetivo principal filtrar e sugerir itens relevantes para os usuários, considerando suas preferências e histórico de interação. Nesse contexto, três abordagens são reconhecidas como fundamentais: Filtragem Baseada em Conteúdo, Filtragem Colaborativa e Métodos Híbridos (RAZA *et al.*, 2024).

2.1.1 Filtragem Baseada em Conteúdo

A Filtragem Baseada em Conteúdo (conhecida em inglês como *Content-Based Filtering*) baseia-se na análise de características inerentes a cada item para gerar recomendações (PAZZANI; BILLSUS, 2007). Em termos práticos, cada produto, serviço ou conteúdo é descrito por um conjunto de atributos (por exemplo, gênero de filme, palavras-chave de um artigo, autor de um livro etc.), enquanto o perfil do usuário reflete as preferências individuais em relação a essas mesmas características.

Inicialmente, destacaram-se métodos que recorriam a modelos de espaço vetorial (por exemplo, medindo similaridade via cosseno), modelos probabilísticos (estimando a probabilidade de o usuário apreciar um item) e algoritmos de árvore de decisão (que agrupam itens com atributos semelhantes) (METEREN, 2000). Com o avanço das técnicas de aprendizado de máquina, surgiram abordagens mais complexas, como redes neurais, resultando em melhores recomendações. De forma complementar, surgiram sistemas inteiramente baseados em GenAI (Inteligência Artificial Generativa), como o de (ZHENG *et al.*, 2023), nos quais LLM são empregados para atribuir um *score* de aderência entre o currículo e a descrição de uma vaga.

Embora essa linha de pesquisa obtenha sucesso em diversas aplicações, ainda enfrenta desafios como a tendência à *overspecialization* (baixa diversidade de recomendações) (BOURGAIS *et al.*, 2022) e o aumento significativo do custo computacional em grandes bases de dados.

2.1.2 Filtragem Colaborativa

A Filtragem Colaborativa (ou *Collaborative Filtering*) parte do princípio de que usuários com gostos parecidos tendem a apreciar itens semelhantes (RAZA *et al.*, 2024). Para isso, constrói-se normalmente uma matriz de interações (explícitas ou implícitas) entre usuários e itens. A partir dessas interações, o sistema consegue prever quais novos itens poderiam interessar a determinado usuário.

Existem dois grandes grupos de técnicas dentro dessa abordagem:

- **Baseadas em memória**: calculam similaridades diretamente a partir dos dados disponíveis (por exemplo, correlação de Pearson ou cosseno) para identificar usuários ou itens semelhantes. Embora sejam intuitivas — como no caso dos algoritmos de vizinhos mais próximos (*k-Nearest Neighbors*) — podem enfrentar problemas de escalabilidade (PARK *et al.*, 2015).
- **Baseadas em modelo**: utilizam métodos como fatoração de matrizes, decompondo a matriz de interações em fatores latentes que representam dimensões de preferência dos usuários e características dos itens (SARWAR *et al.*, 2001).

Mais recentemente, surgiram modelos neurais que elevam a Filtragem Colaborativa a um novo patamar. O (HE, X. *et al.*, 2017), por exemplo, propõe o uso de redes profundas para melhor captar as relações entre usuários e itens. Além disso, métodos que combinam a Filtragem Colaborativa com representações em grafos, como as GNN, também têm demonstrado avanços relevantes ao explorar a estrutura do grafo de interações para modelar padrões mais complexos (WANG *et al.*, 2019).

Apesar de seu bom desempenho em diversos cenários, a Filtragem Colaborativa sofre com o problema de *cold start*, especialmente para usuários ou itens sem histórico de interações, bem como com

a carência de interpretabilidade em modelos mais sofisticados.

2.1.3 Abordagens Híbridas

Dadas as limitações individuais das abordagens anteriores, surgem as chamadas Abordagens Híbridas, cujo objetivo é potencializar as vantagens de cada método, mitigando, ao mesmo tempo, suas fraquezas (BURKE, 2002). Em termos gerais, um sistema híbrido pode ser implementado de diferentes maneiras, como:

- **Combinação ponderada de saídas**: somar as pontuações de um método baseado em conteúdo e de um método colaborativo, atribuindo pesos para cada um conforme sua relevância.
- **Filtragem em cascata**: utilizar um método para pré-filtrar os itens, e outro método para refinar a recomendação entre o subconjunto resultante.
- **Seleção alternada**: empregar heurísticas que definem qual método será mais apropriado em determinada condição, selecionando o resultado de um ou de outro.

As Abordagens Híbridas, portanto, configuram-se como uma alternativa promissora para lidar com cenários em que a escassez de dados e a complexidade do domínio dificultam a aplicação de um único paradigma. Contudo, sua implementação demanda maior cuidado em relação à escolha e integração dos modelos, bem como atenção aos custos computacionais e à interpretabilidade do resultado final (AFOUDI; LAZAAR; AL ACHHAB, 2021).

2.2 MODELOS DE LINGUAGEM DE GRANDE ESCALA

Nos últimos anos, os LLM têm ganhado cada vez mais destaque na área de Processamento de Linguagem Natural. Com base em arquiteturas de *transformers* (VASWANI *et al.*, 2023), esses modelos se caracterizam pela capacidade de aprender representações semânticas ricas a partir de grandes volumes de dados textuais, capturando com maior profundidade os contextos e as nuances linguísticas. Exemplos notáveis incluem o BERT (*Bidirectional Encoder Representations from Transformers*) (DEVLIN *et al.*, 2019) e o GPT-3 (*Generative Pre-trained Transformer 3*) (BROWN *et al.*, 2020), ambos pré-treinados em tarefas como *masked language modeling* ou previsão de próxima palavra, o que lhes confere conhecimento semântico amplo.

2.2.1 Geração de *embeddings* contextuais

Uma das aplicações dos LLM é a geração de *embeddings* contextuais, ou seja, vetores que representam palavras, sentenças ou documentos levando em conta o ambiente em que cada termo aparece. Embora métodos como *Word2Vec* (MIKOLOV *et al.*, 2013) e *GloVe* (BROCHIER; GUILLE; VELCIN, 2019) utilizem o contexto durante o treinamento (por exemplo, por meio de janelas de co-ocorrência), seu resultado final atribui *um único vetor* a cada palavra, independentemente de onde ela ocorra. Em contraste, LLM como BERT ou GPT geram diferentes embeddings para cada ocorrência de uma mesma palavra, refletindo variações de sentido (polissemia) e outras sutilezas linguísticas (DEVLIN *et al.*, 2019; BROWN *et al.*, 2020).

Algumas propriedades dos *embeddings* gerados por esses modelos são:

- **Dimensionalidade e densidade:** os *embeddings* são vetores de tamanho fixo, geralmente variando de centenas a milhares de dimensões. Essa dimensionalidade determina quanto de informação cada vetor pode armazenar: dimensões mais elevadas capturam mais nuances, mas também exigem maiores recursos computacionais. Além disso, são *densos*, diferentemente de representações *one-hot*, em que a maior parte dos valores é zero.

- **Contextualidade:** ao contrário de abordagens estáticas, cada ocorrência de uma palavra, frase ou documento recebe um embedding distinto, levando em conta o texto circundante. Dessa forma, as variações de sentido (por exemplo, "*bank*" como "banco de areia" ou "banco financeiro") são efetivamente capturadas pelo modelo.
- **Similaridade semântica:** os *embeddings* são projetados de modo que palavras ou frases semanticamente próximas resultem em vetores próximos no espaço vetorial. Por exemplo, "*gato*" e "*cachorro*" estarão mais próximos entre si do que de "*carro*". Esse recurso potencializa tarefas como busca semântica, agrupamento e recomendação.
- **Escalabilidade:** *embeddings* gerados pelos LLM podem ser aplicados em grandes bases de dados, sendo calculados em *batches* e armazenados de forma eficiente. Isso os torna apropriados para aplicações em larga escala, como mecanismos de busca e sistemas de recomendação com milhões de usuários e itens.
- **Robustez e adaptabilidade:** esses *embeddings* lidam bem com fenômenos linguísticos complexos, como *polissemia* (múltiplos significados para uma mesma palavra) e *sinonímia* (diferentes termos com sentido semelhante). Além disso, adaptam-se a variados domínios e idiomas, sendo úteis em aplicações *cross-lingual* e *cross-domain*.

2.2.2 Uso em recomendação de empregos

No contexto de sistemas de recomendação voltados a oportunidades profissionais, o uso dos LLM tem se mostrado promissor na atribuição de *scores* de correspondência entre currículos e vagas (ZHENG *et al.*, 2023). Em linhas gerais, esses modelos podem:

- **Analisar descrições de vagas**: extraindo, de modo contextual, requisitos técnicos, comportamentais e níveis de experiência desejados.
- **Codificar currículos**: convertendo qualificações acadêmicas, histórico profissional e competências do candidato em vetores de alta dimensão.
- **Avaliar similaridade semântica**: verificando quão próximo está o perfil do candidato das necessidades da vaga, revelando pontos fortes ou lacunas.

Ao correlacionar diferentes formas de descrever uma mesma habilidade, além de reconhecer termos raros ou específicos de um domínio, os LLM superam as limitações de *embeddings* puramente estáticos. Dessa forma, tornam as recomendações mais precisas e adaptadas a cada cenário profissional, maximizando a chance de compatibilidade entre vaga e candidato (DU *et al.*, 2023).

2.3 REDES NEURAIS DE GRAFOS

2.3.1 Modelagem de Recomendação com Grafos

Grafos são estruturas matemáticas que representam relações entre objetos, sendo formados por um conjunto de vértices (ou nós) e arestas (ou arcos). Formalmente, um grafo pode ser definido como:

$$G = (V, E)$$

onde V é o conjunto de vértices e E o conjunto de arestas. No contexto de recomendação de vagas de emprego, podemos utilizá-los para modelar as interações entre os candidatos e as vagas disponíveis. Dessa forma, cada nó corresponde a um candidato ou a uma vaga, e cada aresta representa a relação de interesse ou candidatura de um usuário em determinada oportunidade.

Para este tipo de problema, a representação pode ser vista de duas maneiras complementares:

- **Grafo Bipartido:** Um *grafo bipartido* é definido como aquele cujos nós podem ser divididos em dois conjuntos disjuntos U e V de modo que toda aresta conecte um nó em U a um nó em V. Em outras palavras, nenhuma aresta conecta dois nós do mesmo conjunto. No exemplo de recomendação de vagas, U poderia representar o conjunto de candidatos, enquanto V seria o conjunto de vagas de emprego. Assim, as arestas surgem apenas entre candidatos e vagas, sem conexões internas em cada grupo.
- **Grafo Heterogêneo:** Além de ser bipartido, esse grafo contém mais de um tipo de nó, tornando-se um *grafo heterogêneo*. Por exemplo, mesmo considerando apenas os candidatos e as vagas, ambos podem apresentar características intrínsecas distintas. Enquanto o nó *Candidato* pode conter atributos como *habilidades*, *experiência* e *formação*, o nó *Vaga* pode trazer informações relacionadas a *cargo*, *requisitos* e *local de trabalho*. Essas diferenças de atributos tornam cada conjunto de nós um tipo distinto, configurando assim um grafo heterogêneo.

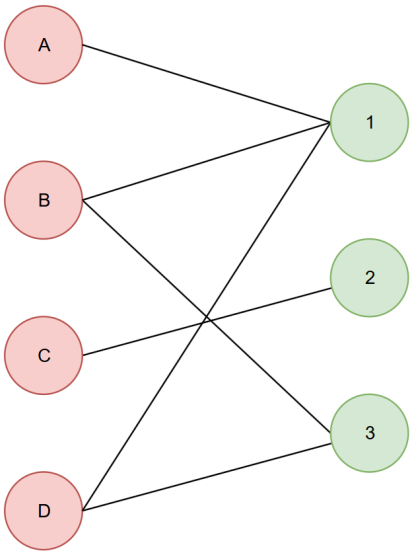

Figura 1 – Exemplo de representação de um sistema de recomendação usando grafos bipartidos. Fonte: Elaborada pelo autor.

Na Figura 1, ilustra-se uma possível representação de um grafo candidato-vaga, em que os conjuntos de nós (sinalizados por cores diferentes) representam tipos distintos (candidatos em verde e vagas em vermelho). Desse modo, o grafo resultante é *bipartido* (pois não há arestas entre nós do mesmo tipo) e *heterogêneo* (uma vez que cada tipo de nó possui atributos próprios e modelam entidades diferentes) (WU, Z. *et al.*, 2021; WU, S. *et al.*, 2022).

2.3.2 Fundamentação Teórica das GNN

As GNN (Redes Neurais de Grafos) são modelos de aprendizado profundo projetados para operar diretamente sobre estruturas de grafos. A ideia central das GNN é a definição de um *grafo computacional*, no qual cada nó aprende uma representação baseada em suas conexões com vizinhos. Essa propagação de informações permite a construção de *embeddings* que incorporam tanto as características individuais dos nós quanto sua posição dentro do grafo (KIPF; WELLING, 2017).

Para ilustrar esse processo, utilizaremos o nó 1 da Figura 1 como exemplo de nó raiz. A partir desse nó, será possível demonstrar como as informações dos vizinhos são agregadas ao longo das camadas de uma GNN, conforme mostrado na Figura 2.

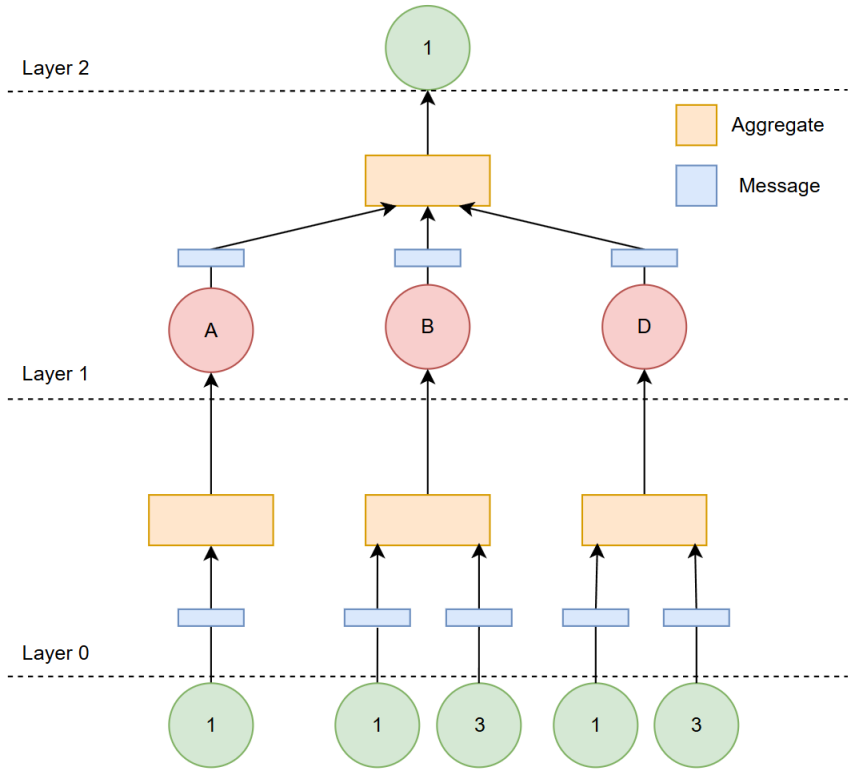

Figura 2 – Exemplo de funcionamento de uma GNN. A camada *Layer 0* representa os nós iniciais do grafo e tem como entrada os atributos destes nós. Já as camadas *Layer 1* e *Layer 2* mostram a propagação e combinação progressiva das informações dos vizinhos, utilizando operações de agregação. Fonte: Elaborada pelo autor.

2.3.3 Propagação de Mensagem

Cada nó gera uma mensagem baseada nas informações de seus vizinhos (GILMER *et al.*, 2017; SCARSELLI *et al.*, 2009). Esse processo pode ser expresso de forma genérica como:

$$m_u^{(k)} = \text{MSG}^{(k)}\left(h_u^{(k-1)}\right)$$

onde $h_u^{(k-1)}$ representa a informação do nó vizinho na camada anterior, e MSG é a função que define a transformação da mensagem. Um exemplo comum é o uso de uma camada linear:

$$m_u^{(k)} = W^{(k)}\, h_u^{(k-1)}$$

onde $W^{(k)}$ é uma matriz de pesos treinável.

2.3.4 Agregação

Após a computação das mensagens, cada nó agrega as mensagens recebidas de seus vizinhos. A agregação pode ser definida como:

$$h_v^{(k)} = \text{AGG}^{(k)}\left(\{m_u^{(k)} \mid u \in \mathcal{N}(v)\},\, m_v^{(k)}\right)$$

onde $\mathcal{N}(v)$ representa o conjunto de vizinhos do nó v. Diferentes estratégias de agregação podem ser utilizadas, como soma, média ou concatenação (KIPF; WELLING, 2017).

2.3.5 Camada da GNN

O conceito fundamental de uma camada de GNN é condensar um conjunto de vetores de entrada (as informações dos vizinhos) em um único vetor de saída (a nova representação do nó). Esse processo,

seguido por uma transformação não linear σ (por exemplo, ReLU), pode ser descrito como:

$$h_v^{(k+1)} = \sigma\Big(W_k \sum_{u \in \mathcal{N}(v)} \frac{h_u^{(k)}}{|\mathcal{N}(v)|} + B_k\, h_v^{(k)}\Big), \quad \forall k \in \{0, 1, \ldots, K-1\}$$

onde B_k é uma matriz de transformação dos atributos do próprio nó.

Esse processo é iterado por K camadas, permitindo que cada nó incorpore informações da vizinhança de K saltos (*k-hops*) do nó raiz (YOU; YING; LESKOVEC, 2021).

2.3.6 Profundidade das Camadas

Um dos principais desafios enfrentados pelas GNN é o fenômeno de *over-smoothing*, que ocorre quando muitas camadas são empilhadas, fazendo com que as representações dos nós se tornem indistinguíveis. Isso reduz a capacidade de diferenciar nós distintos, impactando negativamente tarefas como classificação e recomendação.

À medida que o número de camadas cresce, o *campo receptivo* — o conjunto de nós que contribui para a representação de um nó — também se expande. Em uma GNN com K camadas, cada nó incorpora informações de sua vizinhança de K saltos (*k-hops*), o que pode levar à convergência das representações para valores similares em redes muito profundas.

Algumas estratégias para mitigar esse problema incluem:

- **Conexões residuais**: adicionar conexões *skip* entre camadas para preservar informações originais.
- **Normalização e regularização**: aplicar técnicas como *Batch Normalization* e *Dropout* para manter variação nos embeddings (IOFFE; SZEGEDY, 2015; SRIVASTAVA *et al.*, 2014).
- **Arquiteturas adaptativas**: usar modelos como *Jumping Knowledge Networks* para combinar informações de diferentes profundidades (HE, K. *et al.*, 2015; XU *et al.*, 2018).
- **Ajuste do número de camadas**: limitar a profundidade para evitar que o aumento excessivo do campo receptivo gere convergência indesejada dos embeddings.

2.3.7 Arquiteturas de GNN

2.3.7.1 GCN (Graph Convolutional Network)

Baseando-se na ideia de *mensagem* e *agregação*, a GCN atualiza a representação de cada nó v a partir dos embeddings de seus vizinhos $u \in \mathcal{N}(v)$. Em sua forma simplificada, a equação pode ser escrita como (KIPF; WELLING, 2017):

$$h_v^{(l)} \;=\; \sigma\left(\sum_{u \in \mathcal{N}(v)} W^{(l)}\, \frac{h_u^{(l-1)}}{|\mathcal{N}(v)|}\right),$$

onde:

- **Mensagem**: cada vizinho u gera uma mensagem $m_u^{(l)} = W^{(l)}\, \frac{h_u^{(l-1)}}{|\mathcal{N}(v)|}$, normalizada pelo grau de v.
- **Agregação**: o nó v soma as mensagens recebidas e aplica uma função de ativação σ (por exemplo, ReLU).

2.3.7.2 GraphSAGE (Graph Sample and Aggregate)

Proposto por (HAMILTON; YING; LESKOVEC, 2018), o GraphSAGE mantém a ideia de agregar informações dos vizinhos, permitindo a utilização de diferentes estratégias de agregação (por exemplo,

média, pool ou LSTM) e incorporando a amostragem de vizinhos (*neighbor sampling*) para lidar com grandes grafos. Sua forma simplificada pode ser descrita como:

$$h_v^{(l)} \;=\; \sigma\!\left(W^{(l)}\Big[\, h_v^{(l-1)} \,\|\, \mathrm{AGG}\{\, h_u^{(l-1)} \mid u \in \mathcal{N}(v)\}\Big]\right),$$

onde:

- AGG representa a função de agregação (por exemplo, *mean*, *pool*, *LSTM*) aplicada sobre os embeddings dos vizinhos de v.
- O operador $\|$ denota a concatenação entre o embedding de v e o resultado da agregação dos vizinhos.
- A amostragem de vizinhos reduz a complexidade computacional, tornando o método escalável para grandes grafos.

Além disso, o GraphSAGE propõe a normalização ℓ_2 opcional ao final de cada camada:

$$h_v^{(l)} \leftarrow \frac{h_v^{(l)}}{\|h_v^{(l)}\|_2}, \quad \forall v \in V,$$

garantindo que todos os embeddings tenham a mesma norma, o que pode melhorar a estabilidade e o desempenho.

2.3.7.3 GAT (Graph Attention Network)

Apresentada por (VELIČKOVIĆ *et al.*, 2018), a GAT introduz um mecanismo de atenção que aprende diferentes pesos (α_{vu}) para cada aresta (v, u). Em vez de assumir pesos fixos, cada vizinho u de v recebe um coeficiente de atenção calculado em duas etapas:

1. **Coeficientes de atenção:**

$$e_{vu} = a\Big(\mathbf{W}^{(l)}\,\mathbf{h}_v^{(l-1)},\, \mathbf{W}^{(l)}\,\mathbf{h}_u^{(l-1)}\Big),$$

onde a é uma pequena rede neural (geralmente de camada única) e $\mathbf{W}^{(l)}$ são os pesos treináveis.

2. **Normalização via *softmax*:**

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k\in\mathcal{N}(v)}\exp(e_{vk})},$$

garantindo que $\sum_{u\in\mathcal{N}(v)}\alpha_{vu} = 1$.

Após a normalização, a nova representação do nó v na camada l é dada por:

$$\mathbf{h}_v^{(l)} = \sigma\!\left(\sum_{u\in\mathcal{N}(v)} \alpha_{vu}\,\mathbf{W}^{(l)}\,\mathbf{h}_u^{(l-1)}\right),$$

onde σ é a função de ativação. Esse mecanismo permite atribuir maior importância aos vizinhos mais relevantes, melhorando a capacidade da rede de se adaptar a diferentes estruturas de grafos.

3 MATERIAIS E MÉTODOS

Neste capítulo, são apresentados os materiais e a metodologia adotada para desenvolver o sistema de recomendação de vagas. O objetivo é oferecer uma visão clara dos recursos, dos dados e dos procedimentos que possibilitam a implementação do modelo proposto. Para tanto, o capítulo está organizado da seguinte forma:

1. **Ferramentas e Bibliotecas:** São descritas as tecnologias e os frameworks que sustentam o desenvolvimento do projeto, incluindo modelos de linguagem, bibliotecas de aprendizado de máquina e manipulação de grafos.
2. **Base de Dados:** Detalha-se a origem dos dados, os critérios de filtragem aplicados e a composição das fontes (usuários, vagas e candidaturas) que compõem o grafo.
3. **Construção do Grafo:** Explica-se como o problema de recomendação foi modelado como uma tarefa de predição de arestas, com a definição dos nós (usuários e vagas) e das arestas (candidaturas).
4. **Arquitetura do Modelo:** São descritas as etapas de extração de representações (embeddings) utilizando GNN e o processo de classificação de arestas por meio de operações de similaridade.
5. **Treinamento do Modelo:** Apresentam-se as estratégias de divisão dos dados, amostragem, otimização e monitoramento utilizadas para treinar e validar o modelo, incluindo a técnica de amostragem negativa para gerar exemplos contrastivos.

Cada seção a seguir detalha os procedimentos adotados e justifica as escolhas metodológicas, permitindo uma compreensão aprofundada do fluxo de trabalho empregado para a construção e avaliação do sistema de recomendação.

3.1 FERRAMENTAS

3.1.1 Modelos de Linguagem

- **Transformers**: Fornece acesso a modelos pré-treinados para a extração de embeddings contextuais. Neste trabalho, utilizou-se o modelo gte-v1.5 (ZHANG, X. *et al.*, 2024; LI *et al.*, 2023).

3.1.2 Bibliotecas de Grafos e Aprendizado de Máquina

- **PyTorch Geometric (torch-geometric)**: Utilizada para manipulação de grafos e implementação de GNN.
- **PyTorch**: Biblioteca-base de redes neurais para definição de modelos.
- **scikit-learn**: Empregada no cálculo de métricas de avaliação.
- **pandas** e **NumPy**: Auxiliam na leitura e manipulação de dados, além de fornecerem estruturas eficientes para operações vetorizadas.

3.2 BASE DE DADOS

A base de dados utilizada neste projeto foi obtida a partir do *Job Recommendation Challenge* (HAMNER; WARRIOR; KRUPA, 2012) e dividida em três fontes principais: usuários, vagas de emprego e candidaturas. Após um processo de filtragem (descrito adiante), esses dados resultaram em:

- **Usuários (users.csv)**: **321.231** linhas. Cada registro representa um usuário que efetivamente se candidatou a pelo menos uma vaga, com as seguintes colunas mais relevantes:

UserID: Identificador único de cada usuário.

City, State, Country: Localização (cidade, estado e país).

DegreeType, Major: Informações sobre formação acadêmica (grau e área de estudo).

WorkHistoryCount, TotalYearsExperience: Quantidade de empregos anteriores e total de anos de experiência.

CurrentlyEmployed, ManagedOthers, ManagedHowMany: Dados relativos à situação empregatícia atual (se está empregado, se gerencia equipe e o tamanho dessa equipe).

- **Vagas de Emprego (jobs.csv)**: **365.649** linhas, cada uma referente a uma vaga que recebeu ao menos uma candidatura. Alguns campos importantes são:

  JobID: Identificador único da vaga.

  Title: Título ou cargo associado à vaga.

  Description: Descrição textual das atribuições e responsabilidades.

  Requirements: Requisitos necessários (habilidades, experiência, entre outros).

  City, State, Country: Localização da vaga (cidade, estado e país).

- **Candidaturas (apps.csv)**: **1.603.078** linhas, relacionando usuários e vagas. As colunas são:

  UserID: Identificador do usuário que se candidatou (chave estrangeira para a fonte de usuários).

  JobID: Identificador da vaga à qual o usuário se candidatou (chave estrangeira para a fonte de vagas).

Originalmente, havia uma quantidade maior de registros em cada tabela; porém, aplicou-se um filtro para remover usuários que não se candidataram a nenhuma vaga, bem como vagas que não receberam nenhum aplicante. Assim, cada linha em *apps.csv* corresponde a uma aresta em nosso grafo bipartido (usuário–vaga), enquanto as tabelas *users.csv* e *jobs.csv* fornecem atributos para os nós (por exemplo, localização, formação acadêmica, descrição da vaga).

3.3 CONSTRUÇÃO DO GRAFO

Para modelar o problema de recomendação como uma tarefa de link prediction, estrutura-se um grafo bipartido onde os nós representam **usuários** e **vagas de emprego**, e as arestas correspondem às candidaturas registradas no arquivo *apps.csv*. Para representar esse grafo de forma eficiente e facilitar a implementação de modelos de aprendizado profundo, foi utilizada a estrutura **HeteroData** da biblioteca PyTorch Geometric, que possibilita a modelagem de grafos heterogêneos com diferentes tipos de nós e arestas.

3.4 NÓS USUÁRIOS

- **node_id**: Vetor contendo os identificadores únicos dos usuários.
- **features**: Cada usuário é representado por um embedding obtido a partir da concatenação de três fontes principais de informação:
  - **Formação Acadêmica**: Inclui atributos como Degree Type e Major. Esses dados são processados por um modelo de linguagem (LLM) que gera embeddings contextuais, conforme ilustrado na Figura 3.

- **Experiência Profissional**: Engloba informações numéricas, como WorkHistoryCount, TotalYearsExperience, CurrentlyEmployed, ManagedOthers e ManagedHowMany.
- **Localização**: Dados referentes a City, State, Country são convertidos em coordenadas geográficas por meio da biblioteca geopy.

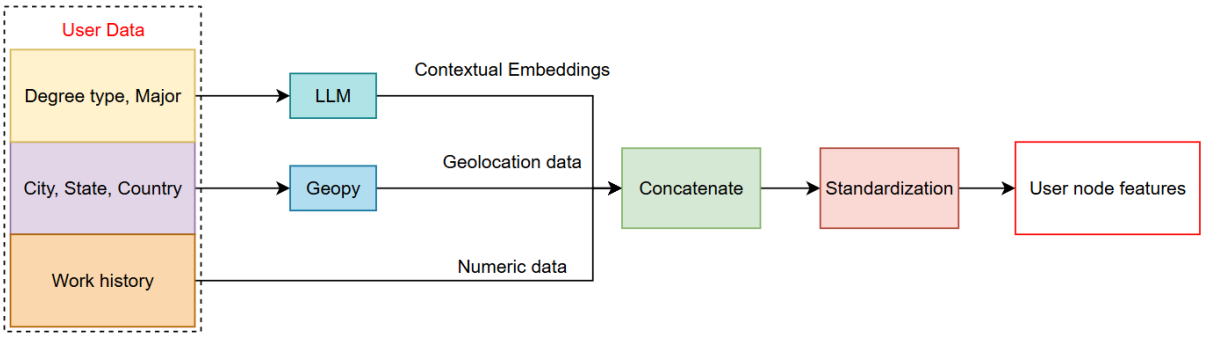

Figura 3 – Fluxo de extração e construção do embedding dos nós de usuários. Fonte: Elaborada pelo autor.

3.5 NÓS VAGAS

- **node_id**: Vetor contendo os identificadores únicos das vagas.
- **features**: Cada vaga de emprego é representada por um embedding obtido a partir da concatenação de:
  - **Informações Textuais**: Engloba campos como Title, Description e Requirements, que são processados por um LLM para gerar embeddings semânticos, conforme ilustrado na Figura 4.
  - **Localização**: Dados referentes a City, State, Country são convertidos em coordenadas geográficas utilizando a biblioteca geopy.

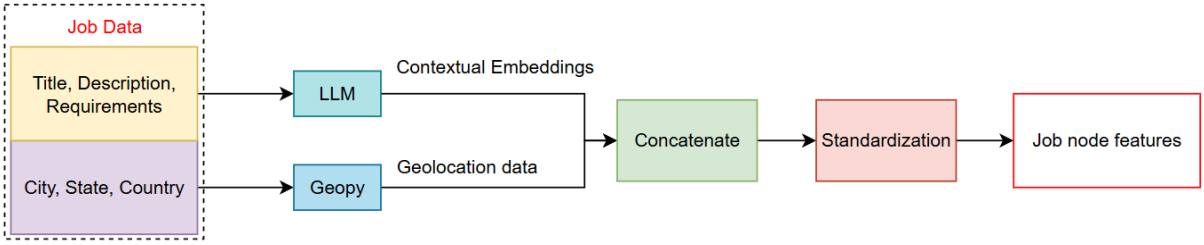

Figura 4 – Fluxo de extração e construção do embedding dos nós de vagas. Fonte: Elaborada pelo autor.

3.6 ARESTAS

Cada linha do arquivo apps.csv define uma aresta que conecta o nó de um usuário (identificado pelo seu user_node_id) ao nó de uma vaga (identificado pelo seu job_node_id). Como se trata de um grafo bipartido, não há conexões entre nós do mesmo tipo, garantindo que todas as relações no grafo correspondam exclusivamente a candidaturas. Na estrutura do grafo, essa relação é representada por uma matriz bidimensional, na qual cada coluna forma um par $(\text{user\_node\_id}, \text{job\_node\_id})$ que modela uma candidatura.

3.7 ARQUITETURA DO MODELO

A previsão de links em grafos baseia-se em duas etapas fundamentais: (i) a extração de representações compactas dos nós (embeddings) por meio de GNN e (ii) a utilização desses embeddings para inferir a existência de conexões entre pares de nós (KIPF; WELLING, 2016; ZHANG, M.; CHEN, 2018). A Figura 5 ilustra o pipeline adotado no modelo.

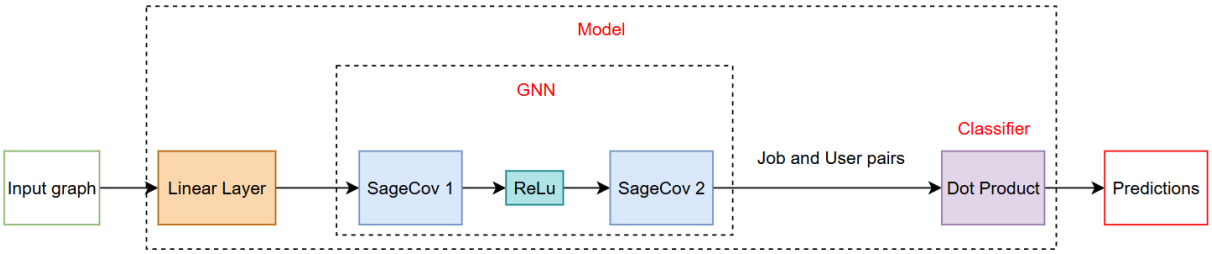

Figura 5 – Diagrama da arquitetura do modelo para previsão de links. Fonte: Elaborada pelo autor.

1. **Extração de Representações com GNN**: Inicialmente, os atributos dos nós do grafo de entrada são processados por meio de uma camada linear. Essa camada é essencial para estabilizar o tamanho das features dos nós, especialmente em grafos heterogêneos, onde os nós podem ter atributos de dimensões diferentes. A camada linear projeta todas as features para um espaço latente de dimensão fixa, permitindo a realização de operações matriciais subsequentes.

   Em seguida, são aplicadas duas camadas GraphSAGE (SAGEConv), que modelam o campo de influência de 2-saltos do nó de origem, característica especialmente útil em sistemas de recomendação, pois possibilita a integração de informações provenientes tanto dos vizinhos diretos quanto dos indiretos.

   Após a primeira camada SAGEConv, aplica-se a função de ativação ReLU (Rectified Linear Unit). Essa função converte todas as entradas negativas para zero, mantendo os valores positivos inalterados, o que introduz a não-linearidade essencial ao modelo. Essa característica permite que a rede capture relações complexas e padrões não lineares nos dados, aumentando sua expressividade. A utilização da camada GraphSAGE se dá por sua capacidade de realizar amostragem eficiente e escalar para grafos de grande porte (GURUKAR *et al.*, 2022).

2. **Predição de Links via Classificação com Produto Interno**: Com os embeddings extraídos, o modelo calcula o produto interno entre os embeddings de um par de nós (por exemplo, usuário e vaga) para quantificar sua similaridade. O produto interno é definido como:

$$s = x_{\text{user}} \cdot x_{\text{job}} = \sum_{i=1}^{n} x_{\text{user},i} \cdot x_{\text{job},i}$$

   onde valores mais altos indicam maior probabilidade de existência de uma conexão (ou candidatura) entre eles.

3.8 TREINAMENTO DO MODELO

Nesta seção, são descritas as estratégias para treinar o modelo de predição de arestas, combinando técnicas de amostragem, otimização e monitoramento.

3.8.1 Divisão dos Dados

A divisão dos dados é realizada por meio da técnica *RandomLinkSplit* do PyTorch Geometric, que particiona o grafo em conjuntos de treinamento, validação e teste de forma aleatória. Essa abordagem transdutiva garante que todos os nós permaneçam disponíveis durante todas as fases, mesmo que algumas arestas sejam removidas para formar os conjuntos de validação e teste, preservando, assim, a estrutura nodal.

No conjunto de treinamento, as arestas são classificadas em duas categorias:

- **Arestas de Mensagem:** São utilizadas para a troca de informações entre os nós durante o *message passing*. Por meio dessas arestas, o modelo propaga características e aprende representações contextuais a partir dos vizinhos.
- **Arestas de Monitoramento:** São empregadas exclusivamente para o cálculo do *loss* durante o treinamento. Embora não participem diretamente da comunicação entre os nós, elas são essenciais para avaliar o desempenho do modelo e orientar o ajuste de seus parâmetros.

No **conjunto de validação**, novas arestas são separadas para o cálculo do *loss*. Entretanto, para garantir a continuidade da troca de informações, todas as arestas que estavam disponíveis no conjunto de treinamento permanecem para o *message passing*. Dessa forma, o modelo utiliza o conhecimento acumulado durante o treinamento enquanto é avaliado em dados que não foram vistos anteriormente.

Por fim, no **conjunto de teste**, o grafo incorpora as arestas utilizadas nas etapas anteriores e adiciona um conjunto final de conexões, servindo como rótulos para a avaliação definitiva

A Figura 6 ilustra visualmente essa divisão, destacando como os diferentes conjuntos de arestas são utilizados em cada fase do processo.

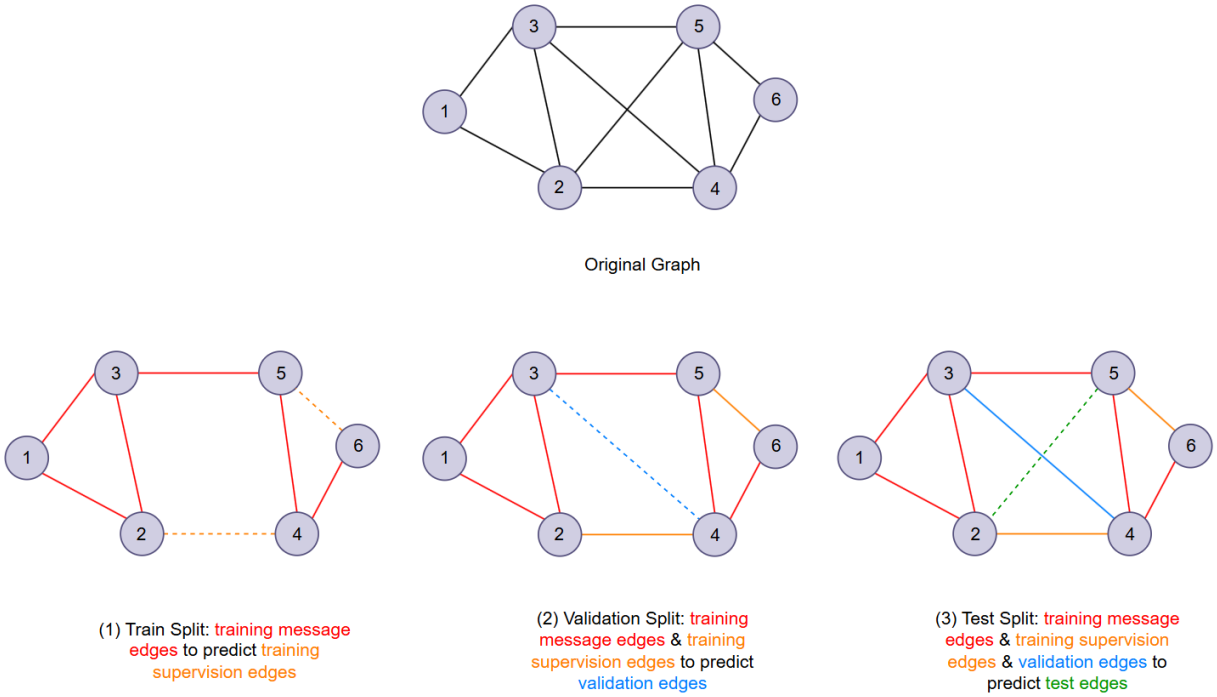

Figura 6 – Esquema da divisão de dados com RandomLinkSplit. Fonte: Elaborada pelo autor.

3.8.2 Amostragem de Vizinhos

Para um treinamento escalável do modelo de predição de arestas, divide-se o grafo em subgrafos menores que podem ser carregados na memória e processados em paralelo. Para isso, emprega-se a técnica de amostragem de vizinhos (HAMILTON; YING; LESKOVEC, 2018), baseada no conceito de

campo receptivo de um nó. A ideia central é que, para calcular o embedding de um nó-alvo, apenas seus vizinhos até k saltos (k-hops) de distância influenciam o cálculo. Por exemplo, em um modelo com duas camadas de GNN, é suficiente armazenar os nós a até 2 saltos do nó raiz para realizar a propagação das informações. Dessa forma, realiza-se a amostragem apenas da vizinhança k-hop do nó de interesse, o que reduz significativamente a complexidade computacional. Além disso, para evitar explosões combinatórias e manter a carga computacional previsível, limita-se a amostragem a, no máximo, H vizinhos por salto, garantindo que o treinamento se mantenha dentro das restrições de memória da GPU (*Graphics Processing Unit*).

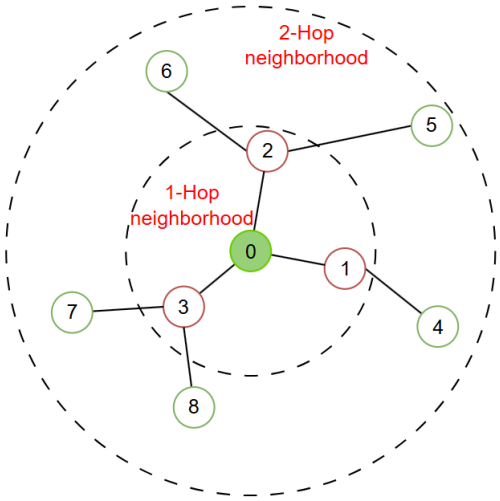

Figura 7 – Representação da vizinhança k-hop de um nó-alvo. Diagrama da arquitetura do modelo para previsão de links. Fonte: Elaborada pelo autor.

3.8.3 Mini-Batch Training

A amostragem de vizinhos gera subgrafos que podem ser organizados em mini-batches, viabilizando treinamento para grafos de grande porte, nos quais não é possível carregar toda a estrutura na memória (GURUKAR *et al.*, 2022).

Na Figura 8, cada nó-alvo é associado a um subgrafo amostrado, formando mini-batches de grafos computacionais independentes. Assim, os embeddings são calculados pela propagação das informações dentro do campo receptivo de cada nó, permitindo o processamento simultâneo de múltiplas instâncias e acelerando o treinamento.

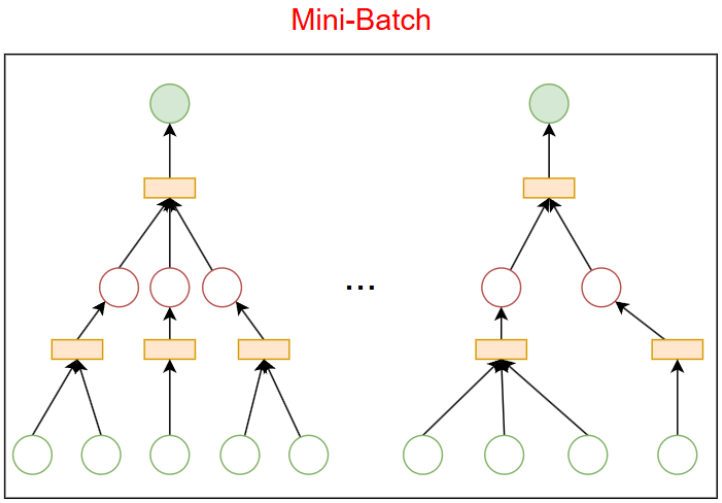

Figura 8 – Esquema de mini-batch training. Fonte: Elaborada pelo autor.

3.8.4 Amostragem Negativa de Arestas

Conforme discutido na Seção 3.6, as arestas do grafo derivam das candidaturas dos usuários às vagas, representando exemplos positivos. No entanto, a ausência de exemplos negativos compromete a capacidade da GNN de diferenciar entre vagas que despertam interesse e aquelas irrelevantes para os usuários. Para mitigar essa limitação, emprega-se a técnica de aprendizagem por contraste (NGUYEN; FANG, 2024; ZHANG, M.; CHEN, 2018), que gera exemplos negativos de maneira controlada. A estratégia consiste em amostrar pares usuário-vaga não conectados, criando arestas negativas balanceadas na proporção 1:1 em relação aos exemplos positivos. Essa abordagem permite que o modelo aprenda a distinguir de forma mais eficaz as classes, o que contribui para uma melhor generalização e robustez na predição de novas conexões.

A Figura 9 ilustra esse processo, onde as arestas contínuas representam conexões positivas, enquanto as arestas tracejadas simbolizam conexões negativas, geradas artificialmente para o treinamento do modelo.

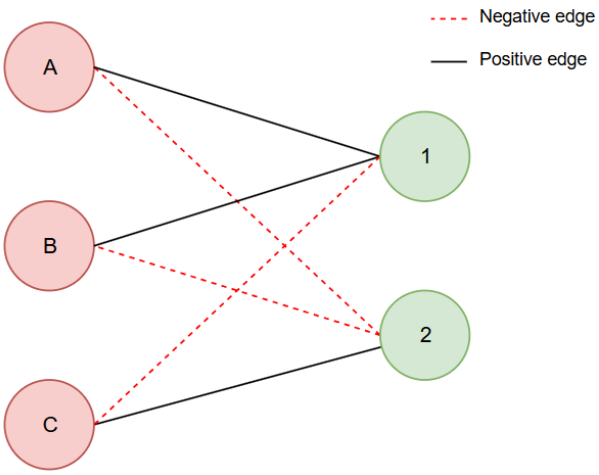

Figura 9 – Representação da amostragem negativa de arestas. Fonte: Elaborada pelo autor. Fonte: Elaborada pelo autor.

3.8.5 Otimização e Monitoramento

A função de perda utilizada é a Cross Entropy with Logits:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i \cdot \log \sigma(z_i) + (1 - y_i) \cdot \log\big(1 - \sigma(z_i)\big)\Big],$$

onde z_i são os logits preditos e y_i os rótulos (1 para positivos, 0 para negativos).

A otimização do modelo é realizada por meio de gradiente descendente, utilizando o otimizador Adam (KINGMA; BA, 2017). Além disso, implementa-se um mecanismo de monitoramento que emprega a técnica de parada antecipada com uma paciência de 10 épocas, interrompendo o treinamento caso a perda de validação não apresente melhora consecutiva. Essa estratégia previne o sobreajuste e otimiza o tempo de treinamento.

4 ANÁLISE EXPERIMENTAL E DISCUSSÃO

Neste capítulo, são apresentados os experimentos realizados para investigar a eficácia do sistema de recomendação proposto. O estudo foi motivado pela necessidade de entender de que forma informações contextuais (extraídas por meio de modelos de linguagem (LLM)) e informações estruturais (obtidas por GNN)) contribuem, individualmente e em conjunto, para a previsão de candidaturas. A partir dessas análises, busca-se responder questões como:

- Qual a relevância dos dados textuais na identificação de padrões de candidatura?
- Em que medida a estrutura do grafo reflete comportamentos colaborativos e padrões de interação?
- A combinação das duas fontes de informação gera ganhos significativos de desempenho?

A seguir, descreve-se a configuração experimental adotada e, posteriormente, são apresentados os resultados obtidos.

4.1 CONFIGURAÇÃO DOS EXPERIMENTOS

Para avaliar a capacidade preditiva do sistema, foram implementadas três abordagens distintas:

1. **LLM-based**: utiliza exclusivamente representações contextuais extraídas de um LLM;
2. **Graph-based**: baseia-se unicamente na estrutura do grafo, explorada por uma GNN;
3. **Híbrida**: integra as abordagens anteriores, combinando informações contextuais (texto) e estruturais (grafo).

Nesta seção, detalhamos os procedimentos adotados para cada abordagem, os critérios de avaliação e as métricas utilizadas para comparar os modelos, a fim de mensurar a contribuição de cada tipo de informação para a predição de candidaturas.

4.1.1 Experimento 1: Abordagem baseada em LLM

Neste primeiro experimento, o modelo se baseia unicamente em representações textuais de usuários e vagas. A Figura 10 ilustra o fluxo adotado:

1. **Extração de *embeddings***: descrições de vagas e perfis de usuários são processados por um LLM, gerando vetores que sintetizam aspectos semânticos de cada entidade.
2. **Formação de pares**: as combinações de usuário e vaga servem como entrada para um classificador MLP, que recebe a concatenação dos *embeddings* correspondentes.
3. **Classificação**: o MLP aprende a distinguir pares positivos (candidaturas efetivas) de pares negativos (ausência de candidatura), retornando a probabilidade de candidatura.

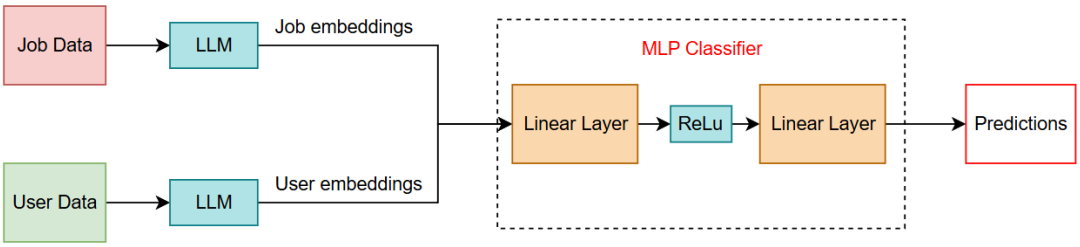

Figura 10 – Arquitetura do modelo baseado em LLM para previsão de candidaturas. (1) Primeira camada linear que concatena a entrada e reduz a dimensionalidade, (2) segunda camada linear que retorna a probabilidade de candidatura. Fonte: Elaborada pelo autor.

4.1.2 Experimento 2: Abordagem baseada em GNN

Neste cenário, um modelo de GNN é treinado sobre um grafo bipartido, porém, sem qualquer informação contextual. Os nós do grafo representam candidatos e vagas, e as arestas indicam candidaturas anteriores. No entanto, os nós não possuem atributos (*nós ocos*), de modo que o modelo depende exclusivamente da estrutura do grafo para inferir novas conexões. Essa configuração é idêntica à descrita no Capítulo 3, diferenciando-se apenas pela remoção das *features* dos nós do grafo.

4.1.3 Experimento 3: Abordagem Híbrida (GNN + LLM)

No terceiro experimento, utiliza-se a abordagem proposta neste trabalho, conforme descrito no Capítulo 3. Dessa maneira, une-se a capacidade de gerar representações contextuais de texto, oriundas das LLMs, com a extração de padrões colaborativos profundos na estrutura do grafo proporcionada pelas GNNs.

4.2 MÉTRICAS DE AVALIAÇÃO

Antes de apresentar as métricas utilizadas, é importante definir os termos relacionados às classificações:

- **TP (True Positive)**: Número de exemplos positivos corretamente identificados.
- **TN (True Negative)**: Número de exemplos negativos corretamente identificados.
- **FP (False Positive)**: Número de exemplos negativos incorretamente identificados como positivos.
- **FN (False Negative)**: Número de exemplos positivos incorretamente identificados como negativos.

Para comparar as três abordagens, foram utilizadas as métricas AUC, Acurácia, *Recall*, *Precision* e *F1-Score*. A seguir, descrevem-se brevemente:

- **Acurácia (Accuracy)**:
$$\text{Acurácia} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$
Mede a proporção de exemplos corretamente classificados (positivos e negativos) em relação ao total.
- **Precisão (Precision)**:
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
Indica quantos dos exemplos preditos como positivos são de fato positivos.
- **Revocação (Recall)**:
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
Mede a fração de casos positivos efetivos que o modelo consegue recuperar.
- **F1-Score**:
$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
Representa a média harmônica entre Precisão e Revocação, equilibrando ambas as métricas.
- **AUC (Área sob a Curva ROC)**: Corresponde à área sob a curva ROC, que relaciona a Taxa de Verdadeiros Positivos (TPR) e a Taxa de Falsos Positivos (FPR). Valores próximos a 1 indicam melhor desempenho.

4.3 RESULTADOS

Os resultados apresentados na Tabela 1 foram obtidos após a execução de cada experimento 100 vezes, utilizando seeds diferentes para o *split* dos dados. A tabela resume as métricas médias e os respectivos desvios padrão das três abordagens avaliadas: Hybrid (GNN + LLM), Graph-based (GNN) e LLM-based (MLP).

Tabela 1 – Resultados das abordagens em termos de AUC, Acurácia, *Recall*, *F1-Score* e *Precision*.

Abordagem	**AUC**	**Accuracy**	**Recall**	**F1-Score**	**Precision**
Hybrid	0.97 ± 0.008	0.90 ± 0.014	0.89 ± 0.019	0.90 ± 0.014	0.91 ± 0.015
Graph-based	0.83 ± 0.001	0.74 ± 0.001	0.66 ± 0.010	0.72 ± 0.003	0.79 ± 0.007
LLM-based	0.72 ± 0.001	0.64 ± 0.003	0.42 ± 0.015	0.54 ± 0.010	0.76 ± 0.007

Observa-se que a abordagem híbrida alcança resultados superiores em todas as métricas, sugerindo que a integração entre informações contextuais (texto) e estruturais (grafo) potencializa a capacidade de previsão de candidaturas.

4.4 DISCUSSÃO DOS RESULTADOS

A análise comparativa entre as três abordagens permite evidenciar o valor complementar das informações textuais e estruturais.

4.4.1 LLM-based (Apenas Contexto)

O modelo que depende exclusivamente de LLM é capaz de capturar elementos como requisitos, habilidades e interesses a partir das descrições de vagas e perfis de usuários, o que o torna flexível em termos de adaptação a variados domínios ou áreas profissionais. Por outro lado, devido à sua ênfase puramente textual, não há conhecimento sobre como os usuários efetivamente interagem uns com os outros e com as vagas, o que limita a identificação de padrões de coaplicação ou de comportamento coletivo. Assim, embora a abordagem seja versátil e mais simples de aplicar em situações com pouca ou nenhuma estrutura de grafo disponível, seu desempenho global é menor em comparação às demais.

4.4.2 Graph-based (Apenas Estrutura)

A abordagem puramente estrutural (GNN) demonstra a importância de explorar padrões topológicos, mesmo na ausência de atributos textuais. O processo de *message passing* propaga informações ao longo das arestas, possibilitando que o modelo identifique subgrupos de nós que compartilham histórico de candidaturas semelhante. Ainda que supere o desempenho do modelo baseado em texto em várias métricas, a GNN sofre com o problema de *cold start*, pois não consegue inferir atributos ou preferências de nós recém-inseridos na rede se estes não possuírem conexões prévias.

4.4.3 Abordagem Híbrida (Contexto + Estrutura)

A abordagem híbrida evidencia a força da fusão entre conteúdo e conectividade. A topologia do grafo ajuda a capturar padrões globais de interação (coaplicações, afinidade entre grupos de vagas e de usuários), enquanto o texto traz detalhes contextuais, como qualificações ou requisitos específicos, refinando as inferências. Essa sinergia se traduz no melhor desempenho em todas as métricas. Por outro lado, o *preprocessamento* necessário para gerar *embeddings* de alta dimensionalidade e o posterior treinamento da GNN podem demandar recursos computacionais elevados, tanto em termos de processamento quanto

de memória, sobretudo em grafos de grande porte. Ainda assim, os resultados finais indicam que esse investimento em capacidade computacional é compensado pela superioridade preditiva do modelo.

5 CONCLUSÃO

Neste trabalho, foram exploradas três estratégias para a previsão de candidaturas: a abordagem **LLM-based**, que utiliza exclusivamente embeddings extraídos de modelos de linguagem; a **Graph-based**, que se apoia na topologia do grafo bipartido; e a **abordagem híbrida**, que integra informações contextuais e estruturais. Os resultados indicaram que a fusão dessas duas fontes – conteúdo textual e conectividade do grafo – permite alcançar desempenho superior em métricas como AUC, Acurácia, Precisão, Revocação e F1-Score, evidenciando o potencial de recomendações mais precisas e personalizadas no contexto de recrutamento e seleção.

5.1 LIMITAÇÕES

Apesar dos avanços obtidos, o modelo proposto apresenta algumas limitações que devem ser consideradas:

- **Dependência de dados de qualidade:** A eficácia da abordagem híbrida está condicionada à disponibilidade de dados textuais ricos e de um grafo suficientemente conectado. Em cenários onde essas condições não são plenamente atendidas, o desempenho do sistema pode ser comprometido.
- **Nós sem arestas:** Em alguns casos, a estrutura do grafo apresenta nós sem conexões, dificultando a propagação de informações. Sem mecanismos adicionais, esses nós ficam desamparados na inferência, impactando a abrangência das recomendações.
- **Pré-processamento caro:** A etapa de pré-processamento dos dados dos nós utilizando a LLM, seguida do treinamento da GNN, pode demandar recursos computacionais significativos, representando um desafio para a escalabilidade da abordagem.

5.2 TRABALHOS FUTUROS

Para superar as limitações apontadas e aprimorar o sistema, sugerem-se as seguintes direções para pesquisas futuras:

- **Modelagem de novo nó de localização:** Desenvolver uma estrutura que incorpore explicitamente dados geográficos, criando nós de localização no grafo. Essa modelagem pode ampliar a capacidade do sistema de identificar padrões regionais e melhorar a personalização das recomendações.
- **Implementação de mecanismos de proxy:** Adotar mecanismos que utilizem o vetor de embeddings inicial para gerar recomendações para nós isolados – aqueles sem arestas. Essa abordagem pode mitigar os efeitos do problema de *cold start*, possibilitando inferências mesmo na ausência de conexões prévias.

5.3 CONSIDERAÇÕES FINAIS

A integração de informações textuais e estruturais demonstrou ser uma estratégia robusta para a previsão de candidaturas. Embora existam desafios – como a dependência de dados de qualidade e o alto custo computacional – os avanços alcançados apontam para um caminho fértil de pesquisa e desenvolvimento. A implementação das melhorias sugeridas poderá não só superar as limitações atuais, mas também potencializar o uso de sistemas de recomendação em contextos complexos de recrutamento e seleção.

REFERENCES

AFOUDI, Yassine; LAZAAR, Mohamed; AL ACHHAB, Mohammed. Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. **Simulation Modelling Practice and Theory**, v. 113, p. 102375, 2021. ISSN 1569-190X.

BENNETT, James; LANNING, Stan. The Netflix Prize. *In*: PROCEEDINGS of KDD Cup and Workshop 2007. [*S.l.*: *s.n.*], 2007.

BERG, Rianne van den; KIPF, Thomas N.; WELLING, Max. **Graph Convolutional Matrix Completion**. [*S.l.*: *s.n.*], 2017. arXiv: 1706.02263 [stat.ML]. Disponível em: https://arxiv.org/abs/1706.02263.

BOURGAIS, Mathieu; ZANNI-MERK, Cecilia; FATALI, Rauf; ALIZADA, Nadir. Avoiding the Overspecialization of Recommender Systems in Tourism with Semantic Trajectories, Initial Thoughts. **Procedia Computer Science**, v. 207, p. 1933–1942, 2022. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022. ISSN 1877-0509.

BROCHIER, Robin; GUILLE, Adrien; VELCIN, Julien. Global Vectors for Node Representations. *In*: THE World Wide Web Conference. [*S.l.*]: ACM, mai. 2019.

BROWN, Tom B. *et al.* **Language Models are Few-Shot Learners**. [*S.l.*: *s.n.*], 2020. arXiv: 2005.14165 [cs.CL]. Disponível em: https://arxiv.org/abs/2005.14165.

BURKE, R. Hybrid Recommender Systems: Survey and Experiments. **User Modeling and User-Adapted Interaction**, v. 12, p. 331–370, 2002.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. [*S.l.*: *s.n.*], 2019. arXiv: 1810.04805 [cs.CL]. Disponível em: https://arxiv.org/abs/1810.04805.

DU, Yingpeng; LUO, Di; YAN, Rui; LIU, Hongzhi; SONG, Yang; ZHU, Hengshu; ZHANG, Jie. **Enhancing Job Recommendation through LLM-based Generative Adversarial Networks**. [*S.l.*: *s.n.*], 2023. arXiv: 2307.10747 [cs.IR]. Disponível em: https://arxiv.org/abs/2307.10747.

GILMER, Justin; SCHOENHOLZ, Samuel S.; RILEY, Patrick F.; VINYALS, Oriol; DAHL, George E. **Neural Message Passing for Quantum Chemistry**. [*S.l.*: *s.n.*], 2017. arXiv: 1704.01212 [cs.LG]. Disponível em: https://arxiv.org/abs/1704.01212.

GURUKAR, Saket; PANCHA, Nikil; ZHAI, Andrew; KIM, Eric; HU, Samson; PARTHASARATHY, Srinivasan; ROSENBERG, Charles; LESKOVEC, Jure. MultiBiSage: A Web-Scale Recommendation System Using Multiple Bipartite Graphs at Pinterest. **arXiv preprint arXiv:2205.10666**, 2022.

HAMILTON, William L.; YING, Rex; LESKOVEC, Jure. **Inductive Representation Learning on Large Graphs**. [*S.l.*: *s.n.*], 2018. arXiv: 1706.02216 [cs.SI]. Disponível em: https://arxiv.org/abs/1706.02216.

HAMNER, Ben; WARRIOR, Road; KRUPA, Wojciech. **Job Recommendation Challenge**. [*S.l.*: *s.n.*], 2012. https://kaggle.com/competitions/job-recommendation. Kaggle.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. **Deep Residual Learning for Image Recognition**. [*S.l.*: *s.n.*], 2015. arXiv: 1512.03385 [cs.CV]. Disponível em: https://arxiv.org/abs/1512.03385.

HE, Xiangnan; LIAO, Lizi; ZHANG, Hanwang; NIE, Liqiang; HU, Xia; CHUA, Tat-Seng. Neural Collaborative Filtering. *In*: PROCEEDINGS of the 2017 IEEE International Conference on Data Mining (ICDM). [*S.l.*]: IEEE, 2017. P. 729–738.

IOFFE, Sergey; SZEGEDY, Christian. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**. [*S.l.*: *s.n.*], 2015. arXiv: 1502.03167 [cs.LG]. Disponível em: https://arxiv.org/abs/1502.03167.

JANNACH, Dietmar; ZANKER, Markus. Value and Impact of Recommender Systems. *In*: RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha (Ed.). **Recommender Systems Handbook**. New York, NY: Springer US, 2022. P. 519–546.

KABBUR, Santosh; NING, Xia; KARYPIS, George. FISM: factored item similarity models for top-N recommender systems. *In*: PROCEEDINGS of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: Association for Computing Machinery, 2013. P. 659–667.

KINGMA, Diederik P.; BA, Jimmy. **Adam: A Method for Stochastic Optimization**. [*S.l.*: *s.n.*], 2017. arXiv: 1412.6980 [cs.LG]. Disponível em: https://arxiv.org/abs/1412.6980.

KIPF, Thomas N.; WELLING, Max. **Semi-Supervised Classification with Graph Convolutional Networks**. [*S.l.*: *s.n.*], 2017. arXiv: 1609.02907 [cs.LG]. Disponível em: https://arxiv.org/abs/1609.02907.

KIPF, Thomas N.; WELLING, Max. **Variational Graph Auto-Encoders**. [*S.l.*: *s.n.*], 2016. arXiv: 1611.07308 [stat.ML]. Disponível em: https://arxiv.org/abs/1611.07308.

KOREN, Yehuda; BELL, Robert; VOLINSKY, Chris. Matrix Factorization Techniques for Recommender Systems. **Computer**, v. 42, n. 8, p. 30–37, 2009.

LI, Zehan; ZHANG, Xin; ZHANG, Yanzhao; LONG, Dingkun; XIE, Pengjun; ZHANG, Meishan. Towards general text embeddings with multi-stage contrastive learning. **arXiv preprint arXiv:2308.03281**, 2023.

LIU, Ping *et al.* **LinkSAGE: Optimizing Job Matching Using Graph Neural Networks**. [*S.l.*: *s.n.*], 2024. arXiv: 2402.13430 [cs.LG]. Disponível em: https://arxiv.org/abs/2402.13430.

METEREN, Robin van. Using Content-Based Filtering for Recommendation. *In.*

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. **Efficient Estimation of Word Representations in Vector Space**. [*S.l.*: *s.n.*], 2013. arXiv: 1301.3781 [cs.CL]. Disponível em: https://arxiv.org/abs/1301.3781.

NGUYEN, Trung-Kien; FANG, Yuan. **Diffusion-based Negative Sampling on Graphs for Link Prediction**. [*S.l.*: *s.n.*], 2024. arXiv: 2403.17259 [cs.LG]. Disponível em: https://arxiv.org/abs/2403.17259.

PARK, Youngki; PARK, Sungchan; JUNG, Woosung; LEE, Sang-goo. Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph. **Expert Systems with Applications**, v. 42, n. 8, p. 4022–4028, 2015. ISSN 0957-4174.

PAZZANI, Michael J.; BILLSUS, Daniel. Content-Based Recommendation Systems. *In*: **The Adaptive Web: Methods and Strategies of Web Personalization**. Edição: Peter Brusilovsky, Alfred Kobsa e Wolfgang Nejdl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. P. 325–341. ISBN 978-3-540-72079-9.

RAZA, Shaina; RAHMAN, Mizanur; KAMAWAL, Safiullah; TOROGHI, Armin; RAVAL, Ananya; NAVAH, Farshad; KAZEMEINI, Amirmohammad. **A Comprehensive Review of Recommender Systems: Transitioning from Theory to Practice**. [*S.l.*: *s.n.*], 2024. arXiv: 2407.13699 [cs.IR]. Disponível em: https://arxiv.org/abs/2407.13699.

RESNICK, Paul; VARIAN, Hal R. Recommender systems. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 40, n. 3, p. 56–58, mar. 1997. ISSN 0001-0782.

SARWAR, Badrul; KARYPIS, George; KONSTAN, Joseph; RIEDL, John. Item-based collaborative filtering recommendation algorithms. *In*: ACM. PROCEEDINGS of the 10th international conference on World Wide Web. [*S.l.*: *s.n.*], 2001. P. 285–295.

SCARSELLI, Franco; GORI, Marco; TSOI, Ah Chung; HAGENBUCHNER, Markus; MONFARDINI, Gabriele. The Graph Neural Network Model. **IEEE Transactions on Neural Networks**, v. 20, p. 61–80, 2009.

SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, v. 15, n. 56, p. 1929–1958, 2014.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. **Attention Is All You Need**. [*S.l.*: *s.n.*], 2023. arXiv: 1706.03762 [cs.CL]. Disponível em: https://arxiv.org/abs/1706.03762.

VELIČKOVIĆ, Petar; CUCURULL, Guillem; CASANOVA, Arantxa; ROMERO, Adriana; LIÒ, Pietro; BENGIO, Yoshua. **Graph Attention Networks**. [*S.l.*: *s.n.*], 2018. arXiv: 1710.10903 [stat.ML]. Disponível em: https://arxiv.org/abs/1710.10903.

WANG, Xiang; HE, Xiangnan; WANG, Meng; FENG, Fuli; CHUA, Tat-Seng. Neural Graph Collaborative Filtering. *In*: PROCEEDINGS of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. [*S.l.*]: ACM, jul. 2019.

WU, Shiwen; SUN, Fei; ZHANG, Wentao; XIE, Xu; CUI, Bin. **Graph Neural Networks in Recommender Systems: A Survey**. [*S.l.*: *s.n.*], 2022. arXiv: 2011.02260 [cs.IR]. Disponível em: https://arxiv.org/abs/2011.02260.

WU, Zonghan; PAN, Shirui; CHEN, Fengwen; LONG, Guodong; ZHANG, Chengqi; YU, Philip S. **A Comprehensive Survey on Graph Neural Networks**. [*S.l.*: *s.n.*], 2021. IEEE Transactions on Neural Networks and Learning Systems, 32(1): 4–24. Disponível em: https://doi.org/10.1109/TNNLS.2020.2978386.

XIE, Xu; SUN, Fei; YANG, Xiaoyong; YANG, Zhao; GAO, Jinyang; OU, Wenwu; CUI, Bin. **Explore User Neighborhood for Real-time E-commerce Recommendation**. [*S.l.*: *s.n.*], 2021. arXiv: 2103.00442 [cs.IR]. Disponível em: https://arxiv.org/abs/2103.00442.

XU, Keyulu; LI, Chengtao; TIAN, Yonglong; SONOBE, Tomohiro; KAWARABAYASHI, Ken-ichi; JEGELKA, Stefanie. **Representation Learning on Graphs with Jumping Knowledge Networks**. [*S.l.*: *s.n.*], 2018. arXiv: 1806.03536 [cs.LG]. Disponível em: https://arxiv.org/abs/1806.03536.

YOU, Jiaxuan; YING, Rex; LESKOVEC, Jure. **Design Space for Graph Neural Networks**. [*S.l.*: *s.n.*], 2021. arXiv: 2011.08843 [cs.LG]. Disponível em: https://arxiv.org/abs/2011.08843.

ZHANG, Muhan; CHEN, Yixin. **Link Prediction Based on Graph Neural Networks**. [*S.l.*: *s.n.*], 2018. arXiv: 1802.09691 [cs.LG]. Disponível em: https://arxiv.org/abs/1802.09691.

ZHANG, Xin *et al.* **mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval**. [*S.l.*: *s.n.*], 2024. arXiv: 2407.19669 [cs.CL]. Disponível em: https://arxiv.org/abs/2407.19669.

ZHENG, Zhi; QIU, Zhaopeng; HU, Xiao; WU, Likang; ZHU, Hengshu; XIONG, Hui. **Generative Job Recommendations with Large Language Model**. [*S.l.*: *s.n.*], 2023. arXiv: 2307.02157 [cs.IR]. Disponível em: https://arxiv.org/abs/2307.02157.