
Trabalho de Conclusão de Curso

# Um Sistema para Ranqueamento e Previsão de Similaridade entre Vagas de Emprego e Candidatos

Hellena Almeida Canuto
hellena.canuto@orion.ufal.br

Orientador:
Prof. Dr. André Luiz Lins de Aquino

Maceió, Janeiro de 2025

Hellena Almeida Canuto

# Um Sistema para Ranqueamento e Previsão de Similaridade entre Vagas de Emprego e Candidatos

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. André Luiz Lins de Aquino

Maceió, Janeiro de 2025

**Catalogação na fonte**
**Universidade Federal de Alagoas**
**Biblioteca Central**
**Divisão de Tratamento Técnico**
Bibliotecária: Helena Cristina Pimentel do Vale – CRB4/661

C871u Canuto, Hellena Almeida.
Um sistema para ranqueamento e previsão de similaridade entre vagas de emprego e candidatos / Hellena Almeida Canuto. – 2025.
35 f.: il.

Orientador: André Luiz Lins de Aquino.
Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) – Universidade Federal de Alagoas, Instituto de Computação. Graduação em Ciência da Computação. Maceió, 2025.

Bibliografia: f. 33-35.

1. Aprendizado de máquina. 2. Redes neurais de grafos. 3. Sistemas de recomendação. 4. Recrutamento laboral. 5. Mecanismo de atenção. I. Título.

CDU: 004.78

*Ao meu avô Alfredo, que me fez prometer que este dia chegaria.*

# Agradecimentos

*"Ainda é cedo, amor. Mal começaste a conhecer a vida."*
– Cartola

Os dias que antecederam a realização deste trabalho foram intensos. Todos os 2087 dias desde que ingressei na graduação o foram, entretanto. Sinto que experimentei todas as fases da vida durante este período de tempo, como em *Into The Wild*, do Sean Penn.

Seis anos que poderiam ter sido apenas quatro. Agora que essa frase me pesa menos, posso dizer que saio da UFAL diferente de como entrei. E ainda bem.

Dito isso, estou procurando as melhores palavras, lapidando e polindo cada uma delas, para um agradecimento à altura de uma família que fez da minha educação a máxima prioridade. Para agradecer aos meus pais, Derlucia e Israel, que subverteram o limite do impossível para levar seus filhos aonde desejavam chegar; a Josefa e Nuia, que nunca mediram esforços para me ajudar quando precisei; e aos meus irmãos, Débora e Israel, que sempre cuidaram de mim e acreditaram em meu potencial. Sem vocês, esta etapa que finalizo seria vazia de significado.

Gostaria também de agradecer ao professor André Aquino, meu orientador e mentor, pelos ensinamentos dentro e fora da universidade; ao Laboratório Orion, minha casa durante a graduação; e ao professor Jorge Artur Coelho pelo incentivo durante a Iniciação Científica.

Aos meus amigos, especialmente Ruanna, Thamara, Thalia, João Victor e Ranna, por terem feito essa jornada mais leve; e ao meu namorado, Geymerson, por sempre se fazer presente para me apoiar em meus objetivos (e pela consultoria *gratuita*). À Gabi, por ter me incentivado a persistir na Iniciação Científica; a Thyago, que, anos atrás, me fez montar um computador e, sem saber, me mostrou o caminho que eu deveria seguir.

E à minha tia Fátima, dona de um coração dos mais generosos e acolhedores, pelo cuidado durante a correria que foram os últimos dias e pelo exemplo de uma vida inteira de seriedade, compromisso e excelência no exercício de sua profissão.

Muito se fala sobre a importância de estar no lugar certo na hora certa, porém, mais do que isso, é preciso estar com as pessoas certas também.

Cada um de vocês é parte disso.

Obrigada, obrigada e obrigada.

*"When you walk through a storm..."*
– Gerry and The Pacemakers

# Resumo

Este trabalho explora o uso de aprendizado de máquina em sistemas de recomendação aplicados ao recrutamento, destacando o potencial de *Graph Attention Networks* (GATs) para modelar relações entre candidatos, vagas e empresas. Um modelo baseado em GATs foi desenvolvido e avaliado em um conjunto de dados real, focando nas tarefas de ranqueamento e predição de similaridade. Os resultados demonstraram desempenho superior do modelo em métricas globais e significativa eficiência computacional, com tempos de execução reduzidos em até quatro vezes no maior subconjunto de dados. Contudo, limitações, como a ausência de informações temporais e a abordagem de amostragem negativa aleatória, indicam oportunidades de aprimoramento. Este estudo contribui para o avanço de sistemas de recomendação no contexto laboral, oferecendo bases para futuras pesquisas que incorporem dados temporais e métodos mais criteriosos de treinamento.

**Palavras-chave**: aprendizado de máquina; redes neurais de grafos; sistemas de recomendação; recrutamento laboral; mecanismo de atenção

# Abstract

This study explores the use of machine learning in recommendation systems applied to recruitment, highlighting the potential of Graph Attention Networks (GATs) to model relationships between candidates, job openings, and companies. A GAT-based model was developed and evaluated on a real dataset, focusing on ranking and similarity prediction tasks. The results demonstrated superior performance in global metrics and significant computational efficiency, with execution times reduced by up to fourfold in the largest dataset subset. However, limitations such as the absence of temporal information and the use of random negative sampling indicate opportunities for improvement. This research contributes to the advancement of recommendation systems in the recruitment context, providing a foundation for future studies that incorporate temporal data and more refined training methods.

**Key-words**: machine learning; graph neural networks; recommendation systems; job recruitment; Graph Attention Networks

# Lista de Figuras

# Lista de Tabelas

# Conteúdo

# 1

# Introdução

Este capítulo introduz o contexto e a justificativa para a pesquisa na Seção 1.1, destacando os desafios e oportunidades relacionados à aplicação de aprendizado de máquina no recrutamento laboral. São apresentados os objetivos geral e específicos do trabalho (Seção 1.2), bem como suas principais contribuições metodológicas e práticas (Seção 1.3). Por fim, na Seção 1.4 descreve-se a estrutura do documento, oferecendo uma visão geral dos capítulos subsequentes.

## 1.1 Contexto e Justificativa

No terceiro semestre de 2024, o Brasil registrou 7 milhões de pessoas desempregadas [IBGE (2025)]. Nesse contexto, o processo manual de busca por vagas e seleção de candidatos é pouco eficiente, tornando-se um desafio significativo tanto para as pessoas que buscam emprego quanto para os recrutadores, que enfrentam um processo demorado e repleto de dificuldades logísticas na identificação e avaliação de perfis.

Um exemplo disso é o serviço de busca de empregos oferecido pelo Sistema Nacional de Emprego (Sine), que leva em média 15 dias para gerar resultados [Governo Federal (2024)]. Para os candidatos, a tarefa de pesquisar vagas, avaliar requisitos e submeter candidaturas exige horas de navegação em plataformas e análise de descrições, muitas vezes repetitivas. Para os recrutadores, a triagem de currículos e a identificação de perfis adequados entre centenas ou milhares de candidatos representa um esforço significativo, com alta demanda de tempo e recursos humanos.

Modelos de aprendizado de máquina [Zhou (2021)] apresentam grande potencial para resolver esse problema, identificando padrões complexos em grandes volumes de dados e oferecendo recomendações mais precisas e personalizadas. Embora aplicações de ML em sistemas de recomendação já tenham sido amplamente exploradas em áreas como e-commerce e entretenimento, sua adoção no contexto de emprego ainda é limitada, especialmente em mercados menos desenvolvidos. A introdução de sistemas baseados em aprendizado de máquina pode mitigar o alto

consumo de tempo associado aos processos de seleção e contratação, automatizando tarefas repetitivas e melhorando a eficiência e a qualidade do recrutamento. Com isso, o tempo necessário para encontrar oportunidades ou identificar talentos ideais é significativamente reduzido.

O uso de *Graph Representation Learning* [Hamilton (2020)] e, mais especificamente, de Redes Neurais de Grafos (GNNs) [Wu et al. (2020)], tem se destacado, nos últimos anos, no contexto de Inteligência Artificial [Russell and Norvig (2002)] aplicada ao recrutamento laboral, pois possibilita a modelagem de relacionamentos entre candidatos, vagas e empresas de forma mais estruturada e contextualizada.

Dentro desse campo, as *Graph Attention Networks* (GATs) emergem como uma técnica de ponta. As GATs, que incorporam mecanismos de atenção para ponderar a importância das conexões entre os nós, permitem uma adaptação mais flexível e precisa aos diferentes contextos dos relacionamentos no grafo. Isso proporciona uma maneira mais eficaz de capturar as interações relevantes entre candidatos, vagas e empresas, levando a melhores recomendações e previsões.

A aplicação de técnicas modernas de aprendizado de máquina nesse domínio tem o potencial de gerar impactos sociais positivos, promovendo maior acessibilidade e igualdade no mercado de trabalho. Nesse sentido, a realização deste estudo justifica-se pela oportunidade de avançar no estado da arte em sistemas de recomendação de emprego, propondo soluções inovadoras que possam beneficiar tanto candidatos quanto empregadores.

## 1.2 Objetivos

### 1.2.1 Objetivo geral

Desenvolver um modelo de ranqueamento e compatibilidade eficaz entre vagas de emprego e candidatos, capaz de prever a similaridade com base no histórico de interesse prévio das empresas e dos candidatos.

### 1.2.2 Objetivos específicos

- Pesquisar e avaliar técnicas de aprendizado de máquina adequadas para sistemas de recomendação no contexto de recrutamento laboral.
- Preprocessar dados reais de recrutamento para o treinamento e validação do modelo.
- Desenvolver um modelo para recomendação.
- Avaliar o desempenho do modelo utilizando métricas apropriadas, como acurácia, precisão e recall.

## 1.3 Contribuições

Este trabalho contribui para o avanço na área de sistemas de recomendação e aprendizado de máquina por meio de três principais inovações. Primeiramente, propõe melhorias nas métricas globais de avaliação de modelos, buscando aprimorar a precisão e a relevância das previsões em sistemas de recomendação. Em segundo lugar, realiza testes extensivos utilizando um dataset real, o que valida a aplicabilidade do modelo em cenários do mundo real, proporcionando resultados mais robustos e realistas. Por fim, ataca a otimização do tempo de execução, implementando abordagens que reduzem significativamente o tempo necessário para treinamento e inferência, tornando o modelo mais eficiente e escalável.

## 1.4 Organização do Trabalho

Este trabalho está organizado em cinco capítulos principais. O Capítulo 2 aborda a caracterização do problema, detalhando sua formulação, com ênfase em ranqueamento e compatibilidade, além de apresentar trabalhos relacionados. No Capítulo 3, é discutido o referencial teórico, incluindo modelos de recomendação, aprendizado de representação de grafos, redes neurais de grafos e métricas de avaliação e similaridade. O Capítulo 4 descreve a metodologia adotada, desde o pré-processamento de dados até a construção do modelo de recomendação, incluindo a geração de embeddings, decodificação, otimização e análise estatística. O Capítulo 5 apresenta os resultados experimentais, discutindo instâncias, ambiente e os resultados obtidos.

# 2

# Caracterização do Problema

Neste capítulo, apresenta-se o problema de pesquisa do trabalho na Seção 2.1, seguido por uma revisão concisa da literatura relevante na Seção 2.2, contextualizando o estudo dentro do campo de investigação.

## 2.1 Formulação do Problema

Sejam $C$ e $V$ os conjuntos de candidatos e vagas, respectivamente, onde cada $c \in C$ e cada $v \in V$ são caracterizados por um conjunto de habilidades, dados educacionais e localização. O objetivo é desenvolver um sistema de recomendação que seja capaz de realizar tanto o ranqueamento das vagas em relação aos candidatos quanto o ranqueamento dos candidatos em relação às vagas, além de identificar correspondências entre candidatos e vagas, levando em consideração as qualificações e requisitos específicos de cada parte.

### 2.1.1 Ranqueamento

O sistema deve ser capaz de ranquear vagas para cada candidato e vice-versa, de acordo com a similaridade entre os perfis das vagas e candidatos.

Para uma entidade $e$ (que pode ser um candidato $c$ ou uma vaga $v$), o sistema deve classificar os itens $i_1, i_2, \ldots, i_n \in I$ (onde $I$ representa o conjunto de vagas $V$ no caso de candidatos, ou o conjunto de candidatos $C$ no caso de vagas) de forma que:

$$\text{rank}(i) \in \mathbb{R} \quad \forall i \in I_e \quad \text{com alta (Acurácia, Precisão, Recall)@k para as recomendações top-K}$$

onde $I_e$ é o conjunto de itens relevantes para a entidade $e$, determinado por fatores como compatibilidade habilidades, requisitos ou localização.

### 2.1.2 Compatibilidade

O sistema deve prever o nível de compatibilidade entre candidatos e vagas, considerando diversos fatores, como habilidades, experiência, educação, preferências de localização e outros requisitos da vaga. O grau de compatibilidade $M(c, j)$ entre o candidato $c$ e a vaga $j$ deve representar o quão bem o candidato se adequa à vaga, levando em conta tanto as qualificações quanto as preferências.

O objetivo é aprender uma função $f : C \times J \to [0, 1]$ tal que o score de compatibilidade predito $M(c, j)$ seja o mais próximo possível da verdadeira adequação do candidato à vaga, que pode ser medida por:

$$f(c, j) \approx \textit{score} \text{ verdadeiro de compatibilidade}(c, j)$$

O verdadeiro *score* de compatibilidade é atribuído como 1 quando há uma candidatura ou contratação, e 0 nos casos em que isso não ocorre.

## 2.2 Trabalhos Relacionados

O aprendizado de máquina tem sido amplamente utilizado para auxiliar processos de recrutamento ao longo dos anos. Por exemplo, Roy et al. (2020) apresentou uma abordagem baseada em classificação de currículos por categorias e recomendação com base na similaridade com descrições de vagas, utilizando o classificador LinearSVM para capturar insights e a semântica dos documentos. Em outra proposta, Ali et al. (2022) desenvolveu uma metodologia automatizada que combina técnicas de aprendizado de máquina e NLP, envolvendo etapas como pré-processamento, extração e representação de atributos, além da construção e avaliação de modelos. Da mesma forma, Mishra et al. (2021) demonstrou a eficácia de um modelo preditivo baseado em variantes do algoritmo K Nearest Neighbours (KNN) para identificar candidatos ideais para vagas na área de tecnologia da informação.

No contexto de Redes Neurais de Grafos aplicadas ao recrutamento e contratação, Yang et al. (2022) introduziu um modelo que melhora a adequação entre candidatos e vagas ao modelar preferências de seleção bidirecionais, considerando perspectivas tanto de empregadores quanto de candidatos. Complementarmente, Sugiarto (2022) tratou a recomendação de emprego como um problema de predição de *links*, aplicando convoluções de grafos para gerar *embeddings* de baixa dimensão que representam vagas e candidatos, facilitando a previsão de candidaturas. Por sua vez, Zhu et al. (2022) demonstrou que grafos heterogêneos e algoritmos de detecção de comunidades podem ajudar estudantes a encontrar cursos e empregos alinhados a seus perfis.

A recomendação de vagas utilizando grafos de conhecimento também tem recebido atenção. Wang et al. (2021) e Wang et al. (2022) exploraram a modelagem do processo de recrutamento

com essa abordagem, enquanto Saito and Sugiyama (2022) mostrou que *knowledge graph embeddings* podem capturar preferências explícitas de candidatos. O estudo evidenciou que incluir comportamentos auxiliares, como favoritar vagas, melhora significativamente o desempenho do ranqueamento. Além disso, Konstantinidis et al. (2022) utilizou técnicas não supervisionadas para extrair habilidades de currículos, destacando a utilidade de representar esses dados em grafos para reduzir erros no processo de extração.

A integração de sistemas de recomendação baseados em grafos de conhecimento com Inteligência Artificial Explicável (XAI)[Minh et al. (2022)] é um campo emergente. Nesse contexto, Upadhyay et al. (2021) e Vultureanu-Albişi et al. (2024) investigaram o uso de Redes de Atenção em Grafos (GATs) para gerar *embeddings* e calcular similaridades entre características de vagas e candidatos, promovendo maior transparência nos sistemas de recomendação.

Por fim, modelos de aprendizado baseados em atenção têm alcançado resultados promissores. Huang et al. (2023) propôs um mecanismo de atenção em redes neurais profundas para capturar relações dinâmicas entre dados textuais e não textuais, otimizando a contribuição de cada aspecto no modelo. Já Mao et al. (2024) integrou redes neurais recorrentes com um mecanismo de atenção em duas camadas, explorando dados heterogêneos para melhorar a personalização e eficácia das recomendações.

# 3

# Referencial Teórico

+

Este capítulo apresenta o referencial teórico utilizado no desenvolvimento deste trabalho, abordando os principais conceitos e técnicas envolvidas. A Seção 3.1 discute modelos de recomendação, com foco na utilização de amostragem negativa. Em seguida, a Seção 3.2 explora o *Graph Representation Learning*, incluindo a definição formal de grafos, predição de *links*, embeddings de nós e a perspectiva *Encoder-Decoder*. A Seção 3.3 detalha as Redes Neurais de Grafos, incluindo propagação de mensagens, convolução aplicada a GNNs e *Graph Attention Networks*. Por fim, a Seção 3.4 aborda as métricas de avaliação e similaridade, como a similaridade cosseno, métricas globais e de ranqueamento.

## 3.1 Modelo de Recomendação

Um modelo de recomendação é um sistema que utiliza dados para prever e sugerir itens relevantes a um usuário, com base em suas preferências, histórico ou características. O objetivo principal é personalizar a experiência do usuário ao reduzir a sobrecarga de escolha e destacar itens que tenham maior probabilidade de interesse ou utilidade.

Os sistemas de recomendação são motores fundamentais em plataformas digitais modernas, e a maneira como captam as preferências do usuário é crucial para sua eficácia. Esses sistemas podem ser classificados em explícitos e implícitos, de acordo com a forma como os dados são coletados e interpretados.

#### Recomendação implícita

A recomendação implícita é uma abordagem que utiliza dados capturados de forma passiva, baseando-se em ações e comportamentos dos usuários para inferir suas preferências. Em vez

de depender de avaliações ou declarações explícitas, esse sistema extrai insights de interações indiretas, oferecendo uma visão mais ampla das inclinações do usuário.

Refere-se ao uso de dados comportamentais coletados automaticamente durante as interações do usuário em uma plataforma. Dentre estes dados estão inclusos candidaturas, contratações, tempo de interação, histórico de compras, entre outros, permitindo que o sistema derive preferências sem exigir um esforço consciente do usuário.

**Recomendação explícita**

A recomendação explícita usa em dados fornecidos ativamente pelos usuários, como habilidades e formação acadêmica, para capturar e refletir suas preferências, tornando-se uma abordagem altamente confiável, mas limitada em termos de escalabilidade. Esses sistemas dependem da interação consciente dos usuários, oferecendo informações diretas que auxiliam na personalização de produtos, serviços ou conteúdos.

Recomendação explícita é uma abordagem que utiliza feedback ativo de usuários para determinar suas preferências e interesses. O sistema depende de ações intencionais, como avaliar um produto, classificar um filme ou responder a uma pesquisa.

### 3.1.1 Uso de Amostragem Negativa

A ideia central da amostragem negativa é selecionar itens que o usuário não interagiu, mas que não são irrelevantes para treinar o modelo. Essa técnica é fundamental para melhorar a capacidade de modelagem de preferências dinâmicas, especialmente quando os dados de interação são esparsos.

Sistemas de recomendação em larga escala frequentemente lidam com milhões de usuários e itens, o que torna a integração de todo o *corpus* de dados no processo de treinamento extremamente custosa em termos computacionais. Embora exista um grande número de usuários e itens, muitos desses sistemas sofrem de um problema de esparsidade de dados, pois a maioria dos usuários interage com apenas uma pequena fração dos itens disponíveis. Além disso, a natureza dinâmica dos interesses dos usuários exige um processo contínuo de adaptação dos modelos de recomendação, que devem se atualizar frequentemente para refletir mudanças nas preferências dos usuários.

Outro desafio significativo é o problema de *cold start*, que ocorre quando novos usuários ou novos itens são introduzidos ao sistema sem dados históricos suficientes para gerar recomendações adequadas. Isso representa um desafio importante para o desempenho dos sistemas de recomendação, já que a falta de interações prévias limita a capacidade do modelo, de prever com precisão as preferências desses novos usuários ou itens.

Para mitigar o problema de esparsidade de dados e facilitar a adaptação contínua às mudanças nas preferências dos usuários, a amostragem negativa tem se mostrado uma técnica essencial

[Ma et al. (2024)]. Ela é usada para modelar as preferências personalizadas dos usuários a partir de exemplos tanto positivos quanto negativos.

#### Amostragem Negativa Uniforme

A Amostragem Negativa Uniforme é uma estratégia de amostragem negativa amplamente utilizada em sistemas de recomendação. Ela consiste em selecionar aleatoriamente itens, a partir das interações não observadas de um usuário, ou seja, itens que o usuário ainda não interagiu. Esses itens não são considerados irrelevantes, mas simplesmente não foram explorados pelo usuário no contexto do sistema de recomendação.

#### Características da Amostragem Negativa Uniforme

- **Seleção Aleatória:** A estratégia escolhe itens aleatórios entre os não observados, sem levar em consideração características específicas dos itens ou do usuário. O objetivo é simplesmente gerar uma amostra negativa de interações não observadas.
- **Facilidade de Implementação:** A principal vantagem dessa abordagem é sua simplicidade e facilidade de implementação, já que não requer cálculos adicionais ou métodos complexos de análise.
- **Diversidade:** Embora a seleção seja aleatória, ela introduz diversidade nas amostras negativas, o que pode ajudar a reduzir o risco de overfitting. Isso ocorre porque o modelo é exposto a uma ampla variedade de itens não observados, o que melhora sua capacidade de generalizar para dados não vistos.

## 3.2 Graph Representation Learning

Graph Representation Learning (GRL) é uma subárea da aprendizagem de máquina que visa mapear grafos e seus elementos (nós, arestas e subestruturas) para representações vetoriais (também chamadas de *embeddings*), preservando as características estruturais e semânticas dos grafos. Essas representações permitem que algoritmos de aprendizado de máquina tradicional sejam aplicados a dados em formato de grafo.

### 3.2.1 Definição Formal de Grafo

Um grafo $G$ é um par ordenado $G = (V, E)$, onde:

- $V$ é o conjunto de nós (ou vértices), representado como $V = \{v_1, v_2, \ldots, v_n\}$, com $n$ sendo o número de nós no grafo.

- $E$ é o conjunto de arestas (ou links), que são pares de nós (ou tuplas de nós), representado como $E = \{(v_i, v_j)\}$, onde $v_i, v_j \in V$.

### 3.2.2 Predição de *Links*

Predição de *links* é uma tarefa de previsão de relações em grafos, com o objetivo de prever arestas faltantes ou ausentes em um grafo, com base nas informações parciais fornecidas. No cenário típico de *relation prediction* (previsão de relações), temos um grafo com um conjunto de nós $V$ e um conjunto de arestas $E$, mas apenas um subconjunto dessas arestas $E_{train}$ está disponível. A tarefa consiste em prever as arestas ausentes $E \setminus E_{train}$, ou seja, identificar quais conexões entre os nós não estão explicitamente representadas, mas provavelmente existem.

#### Definição Formal

Em um grafo $G = (V, E)$, dado um subconjunto de arestas $E_{treino} \subseteq E$, o objetivo da tarefa de predição de *links* é prever o conjunto $E_{teste} = E \setminus E_{treino}$. Isso é feito com base nas informações dos nós e arestas conhecidas, usando estratégias de aprendizado de máquina.

#### Vieses indutivos

Para realizar a predição de links, são necessários vieses indutivos específicos para a natureza dos grafos, como a estrutura local e global de vizinhanças entre os nós. Esses vieses guiam o modelo a aprender como fazer previsões eficazes com base nas propriedades do grafo, como:

- **Vizinhança:** A ideia de que nós próximos uns dos outros têm maior probabilidade de estar conectados.
- **Propriedades do grafo:** Características estruturais, como o grau dos nós, ou a existência de comunidades ou clusters dentro do grafo.

### 3.2.3 *Embeddings* de Nós

*Embeddings* de Nós são representações vetoriais contínuas e de baixa dimensionalidade dos nós de um grafo. O objetivo das *embeddings* é transformar a estrutura discreta de um grafo, onde os nós e arestas são representados como elementos separados, em um espaço contínuo (normalmente vetorial), onde as relações e características dos nós podem ser capturadas de maneira eficiente.

Seu principal objetivo é garantir que a similaridade ou proximidade semântica entre os nós seja preservada no espaço vetorial. Isso significa que nós que estão mais "próximos"ou relacionados no grafo (de acordo com alguma métrica de relacionamento, como vizinhança ou similaridade) devem estar próximos no espaço vetorial.

O processo de *embeddings* pode ser visto como um tipo de aprendizado de representação [Bengio et al. (2013)], onde os nós são mapeados para vetores em um espaço vetorial contínuo. Esses vetores são ajustados durante o treinamento de um modelo para minimizar uma função de perda, com a ideia de que os nós que compartilham estruturas ou propriedades no grafo terão representações vetoriais semelhantes.

### 3.2.4 A Perspectiva Encoder-Decoder

A abordagem Encoder-Decoder no aprendizado de representações de grafos oferece uma perspectiva eficaz para entender a construção de embeddings de nós. Nesse *framework*, a tarefa é dividir o problema de aprendizado de representações gráficas em duas operações principais: o encoder e o decoder.

#### *Encoder*

O *encoder* é responsável por mapear cada nó do grafo em um *embedding*. Formalmente, a operação do encoder é uma função que recebe os nós $v \in V$ (onde $V$ é o conjunto de nós do grafo) e retorna seus respectivos embeddings $z_v \in \mathbb{R}^d$, onde $d$ é a dimensionalidade do vetor. A função pode ser representada da seguinte forma:

$$\text{enc} : V \rightarrow \mathbb{R}^d$$

No caso mais simples, o *encoder* pode realizar uma consulta baseada no ID do nó, onde a função do encoder é definida como:

$$\text{enc}(v) = Z[v]$$

Onde $Z \in \mathbb{R}^{|V| \times d}$ é uma matriz contendo os embeddings de todos os nós, e $Z[v]$ representa a linha de $Z$ que corresponde ao nó $v$. Esse tipo de abordagem, conhecida como shallow embedding, trata o grafo de forma simples e direta, gerando embeddings com base apenas no ID dos nós.

No entanto, o encoder também pode ser mais complexo e incorporar outras informações além do ID do nó, como características dos nós ou a estrutura local do grafo ao redor do nó. Essas versões mais avançadas de encoders são frequentemente chamadas de Redes Neurais de Grafos (Seção 3.3), que podem aprender representações de nós levando em consideração suas conexões e características contextuais.

#### *Decoder*

O papel do *decoder* é usar os *embeddings* gerados pelo encoder para reconstruir informações sobre o grafo. O *decoder* tenta prever certas estatísticas ou relações gráficas a partir dos *embed-*

*dings* dos nós. Por exemplo, dado o *embeddings* $z_u$ de um nó $u$, o *decoder* pode tentar prever o conjunto de vizinhos de $u$, ou seja, a vizinhança local do nó.

Em um *decoder pairwise* (decodificador baseado em pares), a função do *decoder* pode ser representada por:

$$\text{dec} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$$

Esse tipo de *decoder* prevê a relação ou similaridade entre dois nós com base em seus embeddings. Por exemplo, um *decoder* simples poderia prever se dois nós são vizinhos no grafo. Aplicando o *decoder* a um par de embeddings $(z_u, z_v)$, o resultado seria uma reconstrução da relação entre os nós $u$ e $v$.

O objetivo do sistema é minimizar a perda de reconstrução, ou seja, garantir que a relação entre os nós prevista pelo *decoder* seja a mais próxima possível da relação real entre os nós no grafo. O modelo tenta otimizar para que:

$$\text{dec}(\text{enc}(u), \text{enc}(v)) \approx S[u, v]$$

Onde $S[u, v]$ é uma medida de similaridade entre os nós $u$ e $v$, que pode ser, por exemplo, uma métrica baseada em vizinhança (como o valor da matriz de adjacência $A[u, v]$, que indica se os nós são ou não vizinhos).

A minimização da perda de reconstrução envolve aprender os embeddings de forma que as relações no grafo (seja vizinhança direta ou algum outro tipo de relação) sejam corretamente refletidas nas representações vetoriais dos nós. Isso pode ser feito utilizando diversas métricas de similaridade ou sobreposição de vizinhanças entre os nós, dependendo da aplicação.

### 3.2.5 Grafos de Conhecimento

Grafos de conhecimento (ou *Knowledge Graphs*, em inglês) são uma representação estruturada de informações que captura relações entre entidades (como pessoas, lugares, conceitos, objetos, etc.) e as propriedades dessas entidades, bem como as interações entre elas. Esses grafos são utilizados para organizar, representar e consultar grandes volumes de dados complexos de uma forma que imite a forma como os humanos compreendem e interagem com o conhecimento. A Figura 3.1 exemplifica um grafo de conhecimento simples.

#### Definição Formal

Um grafo de conhecimento é definido como uma tripla $G = (V, E, R)$, onde:

- $V$ é o conjunto de nós (ou vértices), representando entidades ou conceitos. Cada nó $v \in V$ pode ser associado a um conjunto de atributos $A(v)$, onde:

$$A(v) = \{(k, v_k) \mid k \in \text{Domínios de atributos}, v_k \in \text{Valores possíveis}\}.$$

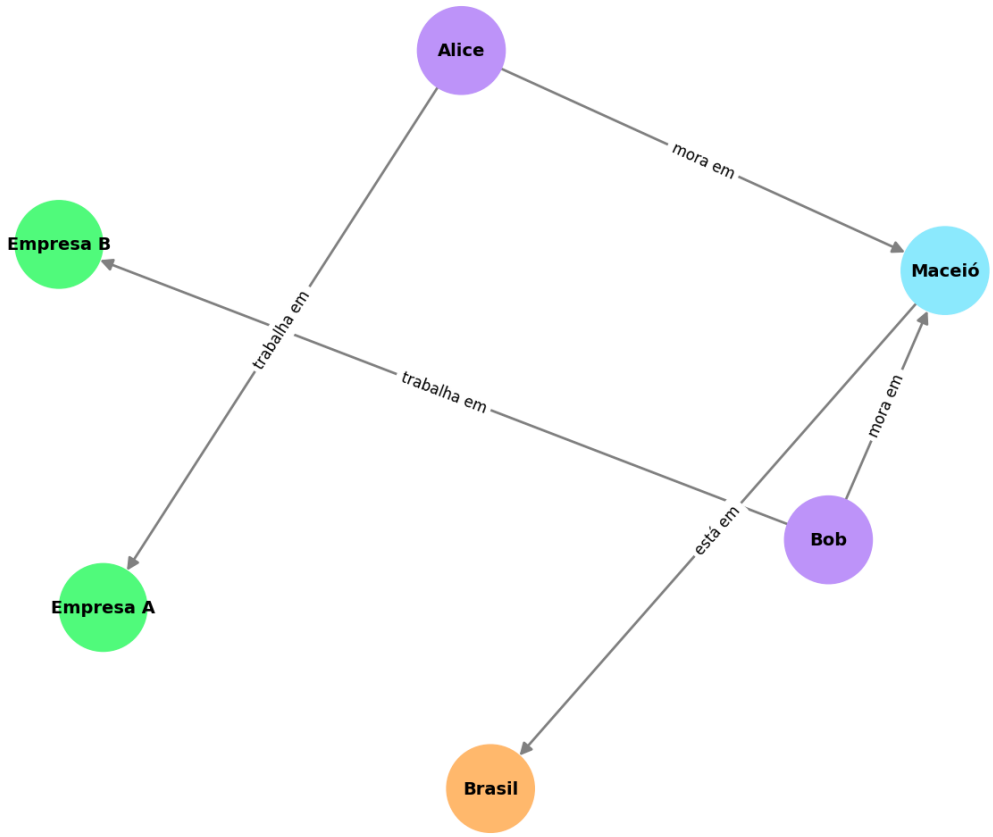


Figura 3.1: Exemplo de Grafo de Conhecimento. Fonte: autor.

- $E$ é o conjunto de arestas direcionadas, representando relações entre os nós. Cada aresta $e \in E$ é definida como um triplo:

  $$e = (h, t, r),$$

  onde:

  - $h \in V$ é o nó origem,
  - $t \in V$ é o nó destino,
  - $r \in R$ é o tipo da relação associada à aresta.

- $R$ é o conjunto de tipos de relações, que define as possíveis relações entre os nós. Cada relação $r \in R$ é uma função semântica que descreve o vínculo entre $h$ e $t$.

**Propriedades**

- **Direcionado:** A direção das arestas captura o sentido semântico da relação $r$.
- **Rotulado:** Os nós e as arestas possuem rótulos ou identificadores únicos.

- **Multigrafo:** Pode haver múltiplas relações entre os mesmos pares de nós, cada uma com um tipo $r \in R$.

## 3.3 Redes Neurais de Grafos

Redes Neurais de Grafos são um tipo de rede neural projetada para lidar com dados estruturados como grafos, onde as informações estão organizadas em nós (vértices) e arestas (conexões). Em contraste com redes neurais tradicionais, que operam em dados vetoriais ou matriciais, GNNs foram criadas para aprender representações de grafos, explorando as relações entre nós e suas conexões para realizar tarefas como classificação de nós, previsão de links, e análise de grafos.

### 3.3.1 Propagação de Mensagens

O processo de propagação de mensagens em Redes Neurais de Grafos é o mecanismo fundamental que permite que as redes neurais operem em grafos, trocando e atualizando informações entre os nós do grafo. O modelo de GNN, baseado nesse framework de propagação de mensagens, é projetado para aprender representações (embeddings) dos nós, levando em consideração não apenas suas características próprias, mas também as características de seus vizinhos no grafo.

Durante cada iteração de propagação de mensagens em uma GNN, o *embedding* oculto $h_u^{(k)}$ de um nó $u$ é atualizado com base nas informações agregadas dos vizinhos de $u$, denotados por $N(u)$ (o conjunto de nós vizinhos de $u$).

Este processo de atualização pode ser formalizado da seguinte maneira:

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\left(h_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\{h_v^{(k)} \mid v \in N(u)\}\right)\right)$$

Aqui:

- AGGREGATE é uma função que toma as embeddings dos vizinhos de $u$ e gera uma "mensagem"$m_{N(u)}^{(k)}$ com base nessas informações.

- UPDATE é uma função que combina essa mensagem $m_{N(u)}^{(k)}$ com a *embeddings* anterior de $u$ (ou seja, $h_u^{(k)}$) para gerar a *embeddings* atualizada $h_u^{(k+1)}$.

**Processo de Propagação de Mensagens**

1. **Inicialização:** No início (quando $k = 0$), as embeddings de todos os nós são definidas pelas características de entrada $x_u$ de cada nó $u$. Ou seja:

$$h_u^{(0)} = x_u \quad \text{para todo } u \in V$$

2. **Agregação:** Em cada iteração $k$, a função AGGREGATE coleta as embeddings dos vizinhos $h_v^{(k)}$ de cada nó $v \in N(u)$ e realiza uma operação de agregação para gerar uma mensagem $m_{N(u)}^{(k)}$. A operação de agregação pode ser uma soma, média, ou qualquer outra função diferenciável.

3. **Atualização:** Após a agregação, a função UPDATE combina a mensagem $m_{N(u)}^{(k)}$ com a *embeddings* atual $h_u^{(k)}$ do nó $u$, gerando a nova *embeddings* $h_u^{(k+1)}$. Isso representa o processo de atualização de informações de um nó com base no que seus vizinhos "passaram"para ele.

4. **Iteração:** O processo de agregação e atualização é repetido por $K$ iterações. O número de iterações $K$ controla até que ponto as informações de vizinhos distantes são propagadas através do grafo.

5. **Resultado Final:** Após $K$ iterações, a *embeddings* final $h_u^{(K)}$ de cada nó $u$ é obtida. Essa *embeddings* pode ser usada para diversas tarefas, como classificação de nós, previsão de links, entre outros. A *embeddings* final também é chamada de representação de nó $z_u$, dado por:

$$z_u = h_u^{(K)} \quad \text{para todo } u \in V$$

#### Propriedade de Equivalência por Permutação

Uma característica importante das GNNs definidas dessa forma é que elas são equivariantes por permutação. Isso significa que, independentemente de como os nós são ordenados ou rotulados, o modelo de GNN vai produzir os mesmos resultados. Isso é garantido porque a função AGGREGATE opera em um conjunto de nós, e não na ordem específica em que os nós são apresentados. Ou seja, a troca de posição dos nós vizinhos não altera o comportamento da GNN.

#### Equivalência a Propagação de Mensagens

O processo de propagação de mensagens pode ser entendido como uma generalização de propagação de crenças (*belief propagation* [Yedidia et al. (2003)], uma técnica tradicional para inferência em grafos probabilísticos. Na GNN, a propagação de mensagens ocorre de forma diferenciável, permitindo otimizar as embeddings dos nós utilizando técnicas de aprendizado supervisionado ou não supervisionado.

### 3.3.2 Convolução aplicada à GNNs

A convolução é uma operação matemática que aplica um filtro (ou kernel) em partes locais de uma entrada (como uma imagem, um sinal ou um texto). O objetivo é extrair características

relevantes, como bordas, texturas ou padrões, que são usadas em camadas posteriores para realizar tarefas como classificação ou segmentação.

No contexto de GNNs, uma camada convolucional é um mecanismo de transformação que opera em grafos, análogo às camadas convolucionais usadas em redes neurais convolucionais [Li et al. (2021)]. No contexto de GNNs, a ideia é generalizar a convolução para dados não euclidianos, como grafos, onde a topologia do grafo define a estrutura de vizinhança entre os nós. A camada convolucional em GNNs é responsável por agregar e processar as informações dos vizinhos de cada nó, permitindo que o modelo aprenda representações significativas dos nós e das arestas do grafo.

Como em grafos, os nós estão conectados de forma arbitrária, ou seja, as conexões entre os nós não seguem uma estrutura regular (como *pixels* em uma imagem). As camadas convolucionais em GNNs têm como objetivo capturar as dependências estruturais desses nós e suas conexões, propagando informações ao longo do grafo de maneira eficiente. Esse processo de "propagação"e "agregação"de informações é uma forma de adaptar as operações convolucionais tradicionais para grafos, permitindo que o modelo extraia informações de contexto local de maneira adaptável, sem a necessidade de uma grade ou estrutura regular.

**Definição Geral**

A operação de uma camada convolucional em uma GNN pode ser expressa da seguinte forma:

$$h_u^{(k+1)} = \sigma\left(\sum_{v \in N(u)} \frac{1}{c_{uv}} \cdot W^{(k)} h_v^{(k)}\right)$$

Onde:

- $h_u^{(k)}$ é a representação do nó $u$ na camada $k$.
- $N(u)$ é o conjunto de vizinhos do nó $u$ (incluindo o próprio nó em alguns casos).
- $W^{(k)}$ é uma matriz de pesos para a camada $k$.
- $c_{uv}$ é um fator de normalização que pode ser usado para ajustar a contribuição dos vizinhos (por exemplo, o grau do nó).
- $\sigma$ é uma função de ativação não-linear (como ReLU ou sigmóide).

### 3.3.3 *Graph Attention Networks*

As *Graph Attention Networks* são uma extensão poderosa das Redes Neurais de Grafos que introduzem o conceito de atenção no processo de agregação das mensagens entre nós. Trata-se de uma abordagem baseada em convolução que permite que diferentes vizinhos de um nó

contribuam de forma diferente para a atualização da representação desse nó, atribuindo pesos de atenção a essas conexões.

Em modelos tradicionais de GNNs, todos os vizinhos de um nó contribuem igualmente para sua atualização, ou seja, a agregação das mensagens é feita de forma homogênea. No entanto, nem todos os vizinhos são igualmente relevantes para o nó. Em muitas situações, alguns vizinhos podem fornecer informações mais valiosas do que outros. A GAT introduz o mecanismo de atenção para resolver esse problema, permitindo que o modelo aprenda quais vizinhos têm mais impacto na atualização de um nó, o que leva a representações mais informativas.

### Estrutura e Funcionamento

O funcionamento das GATs pode ser descrito em três componentes principais:

1. **Cálculo dos Coeficientes de Atenção:** Para cada aresta $(u, v)$ entre dois nós $u$ e $v$, calculamos um coeficiente de atenção $\alpha_{uv}$, que indica a importância de $v$ para o nó $u$. Isso é feito aplicando uma função de atenção a um vetor que combina as representações dos dois nós envolvidos. A ideia é que o modelo aprenda, durante o treinamento, quais vizinhos têm maior relevância para cada nó.

   O cálculo do coeficiente de atenção geralmente envolve duas etapas:

   - Concatenar as representações dos nós $u$ e $v$ (ou aplicar uma transformação linear sobre elas).
   - Passar essa concatenação por uma função de atenção, tipicamente uma rede neural simples, que gera um escalar que indica a importância relativa de $v$ para $u$.

   A atenção é muitas vezes normalizada usando uma função softmax, para garantir que os coeficientes de atenção somem 1.

   A fórmula para calcular o coeficiente de atenção entre dois nós $u$ e $v$ pode ser expressa como:

   $$\alpha_{uv} = \text{softmax}\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \parallel \mathbf{W}\mathbf{h}_v]\right)\right)$$

   Onde:

   - $\mathbf{h}_u$ e $\mathbf{h}_v$ são as representações dos nós $u$ e $v$, respectivamente.
   - $\mathbf{W}$ é uma matriz de pesos treinável.
   - $\parallel$ denota a concatenação das representações dos nós.
   - $\mathbf{a}$ é um vetor de pesos de atenção, que é aprendido durante o treinamento.
   - LeakyReLU é uma função de ativação que ajuda a evitar problemas com valores negativos.

2. **Agregação com Atenção:** Uma vez que os coeficientes de atenção $\alpha_{uv}$ são calculados, a atualização da representação de um nó $u$ é feita como uma agregação ponderada das representações de seus vizinhos $v$, usando os coeficientes de atenção como pesos. Ou seja, cada vizinho $v$ contribui para a representação de $u$ de acordo com a importância aprendida $\alpha_{uv}$.

   A agregação pode ser expressa como:

$$h_u^{\text{new}} = \sigma\left(\sum_{v \in N(u)} \alpha_{uv} \mathbf{W}\mathbf{h}_v\right)$$

   Onde:

   - $N(u)$ é o conjunto de vizinhos de $u$.
   - $\mathbf{W}$ é a matriz de pesos aplicada à representação de cada vizinho.
   - $\sigma$ é uma função de ativação não-linear (como ReLU ou tanh).

3. **Multiplicação de Várias Cabeças de Atenção:** Uma característica importante das GATs é o uso de múltiplas cabeças de atenção. Isso significa que, em vez de usar uma única atenção para calcular a importância dos vizinhos, o modelo usa várias atenções paralelas (ou "cabeças"), e depois as combina. Isso permite que a rede aprenda diferentes aspectos da relação entre os nós e melhora a expressividade do modelo.

   A atualização final de um nó $u$ pode ser expressa como uma combinação das atualizações de várias cabeças de atenção:

$$h_u^{\text{new}} = \|_k \sigma\left(\sum_{v \in N(u)} \alpha_{uv}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_v\right)$$

   Onde $\|_k$ denota a concatenação das saídas de $K$ cabeças de atenção.

## 3.4 Métricas de Avaliação e Similaridade

### 3.4.1 Similaridade Cosseno

A similaridade cosseno é uma medida usada para calcular a semelhança entre dois vetores em um espaço vetorial, com base no ângulo entre eles, independentemente de sua magnitude. Em outras palavras, a similaridade cosseno mede o quão próximos dois vetores estão em termos de direção, e não em termos de comprimento.

#### Definição Matemática

A fórmula para calcular a similaridade cosseno entre dois vetores **A** e **B** é:

$$\text{similaridade cosseno}(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Onde:

- $\mathbf{A}\cdot\mathbf{B}$ é o produto escalar entre os vetores **A** e **B**.
- $\|\mathbf{A}\|$ e $\|\mathbf{B}\|$ são as normas (ou magnitudes) dos vetores **A** e **B**, calculadas como:

$$\|\mathbf{A}\| = \sqrt{\sum_{i=1}^{n} A_i^2}$$

$$\|\mathbf{B}\| = \sqrt{\sum_{i=1}^{n} B_i^2}$$

#### Propriedades

1. **Escala Invariante:** A similaridade cosseno não é afetada pela magnitude dos vetores. Ou seja, dois vetores que apontam na mesma direção terão uma similaridade cosseno de 1, independentemente do comprimento dos vetores.
2. **Mede a Similaridade Direcional:** Ela avalia a semelhança com base na direção dos vetores, o que a torna útil em muitas tarefas, como análise de texto, onde queremos comparar a semelhança de significado e não a magnitude dos vetores.

### 3.4.2 Métricas Globais

As métricas acurácia, precisão e recall globais são amplamente usadas em tarefas de classificação para avaliar a performance geral de um modelo com base na matriz de confusão.

#### Acurácia

A acurácia mede a proporção de previsões corretas (tanto positivas quanto negativas) em relação ao total de previsões.

$$\text{Acurácia} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$$

Onde:

- **TP (*True Positives*):** Previsões corretas de casos positivos.

- **TN (*True Negatives*):** Previsões corretas de casos negativos.
- **FP (*False Positives*):** Previsões incorretas de casos negativos como positivos.
- **FN (*False Negatives*):** Previsões incorretas de casos positivos como negativos.

#### Precisão

A precisão mede a proporção de exemplos classificados como positivos que realmente são positivos. Ela avalia a exatidão do modelo em prever casos positivos.

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

#### Recall

O recall mede a proporção de exemplos positivos que foram corretamente identificados pelo modelo. Ele avalia a capacidade do modelo de recuperar todos os casos positivos.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

### 3.4.3 Métricas de Ranqueamento

As métricas de avaliação de ranqueamento, como Acurácia@k, Precisão@k e Recall@k, são usadas em sistemas de recomendação e tarefas de ranqueamento para avaliar o desempenho com base no "top-k"itens recomendados.

#### Acurácia@k (A@k)

Avalia se pelo menos um dos itens relevantes aparece entre os k itens recomendados.

$$\text{Acurácia@k} = \frac{\text{Números de vezes que itens relevantes aparecem no top-k}}{\text{Número total de consultas}}$$

#### Precisão@k (P@k)

Avalia a proporção de itens relevantes entre os k itens recomendados, i. e., o quão relevante é a lista de recomendações.

$$\text{Precisão@k} = \frac{\text{Número de itens relevantes no top-k}}{k}$$

**Recall@k (R@k)**

Avalia a proporção de itens relevantes recuperados entre todos os itens relevantes existentes, i. e., mede a capacidade do sistema de recuperar os itens relevantes.

$$\text{Recall@k} = \frac{\text{Número de itens relevantes no top-k}}{\text{Número total de itens relevantes}}$$

# 4

# Metodologia

Este capítulo descreve a metodologia adotada para o desenvolvimento do modelo experimental, dividida em etapas essenciais. A Seção 4.1 aborda o pré-processamento dos dados, detalhando as etapas de limpeza e transformação dos dados brutos. A Seção 4.2 explica a representação dos dados, destacando como as informações são estruturadas para alimentar o modelo. A Seção 4.3 descreve o modelo de recomendação, incluindo a geração de embeddings, o processo de decodificação e recomendação, e as técnicas de otimização e treinamento. Por fim, a Seção 4.4 apresenta a análise estatística utilizada para avaliar o desempenho do modelo, garantindo a robustez e a validade dos resultados obtidos.

## 4.1 Pré-Processamento de Dados

A partir de um conjunto de dados contendo informações sobre candidatos e vagas, são extraídos os identificadores únicos de cada entidade, as habilidades declaradas pelos candidatos, último nível de formação educacional e o município de residência. De maneira semelhante, das vagas são extraídos os identificadores únicos, as habilidades requeridas, o município onde estão localizadas e o nível de escolaridade exigido.

Como, com exceção dos identificadores únicos, os dados extraídos são textuais, estão sujeitos a inconsistências e erros de digitação. Por exemplo, uma mesma habilidade pode ser registrada de maneiras diferentes, como "Gerência"e "Gerenciamento". Para lidar com essas variações, os dados passaram por processos de normalização (incluindo conversão para minúsculas, remoção de stopwords e símbolos, lemmatização). Essas etapas são fundamentais para unificar entidades descritas de forma textual distinta e reduzir o volume de dados a serem processados, otimizando o desempenho do treinamento.

Uma vez processados, são geradas tabelas contendo os identificadores únicos de vagas, candidatos, habilidades e níveis educacionais. Além disso, são criadas tabelas que relacionam

cada candidato e cada vaga às suas respectivas habilidades, nível educacional e município. Por exemplo, um candidato com habilidades como "Análise de Dados"e "SQL"seria associado a essas competências específicas, juntamente com seu nível educacional, como "Graduação", e seu município de residência. Esse processo organiza as informações de forma estruturada e consistente.

Como elucidado na Seção 3.1.1, técnicas de amostragem negativa podem ser aplicadas para evitar que o modelo seja enviesado por amostras pouco informativas. Dessa forma, é aplicada a técnica de Amostragem Negativa Uniforme e são selecionadas todas as amostras positivas disponíveis, uma vez que estão em menor número, e um número igual de amostras aleatórias negativas para compor o conjunto de possíveis arestas do modelo.

## 4.2 Representação

As tabelas são então utilizadas para construir um grafo de conhecimento (Figura 4.1), no qual candidatos, vagas, habilidades, níveis de educação e municípios de localização são representados como nós, enquanto as relações entre essas entidades são representadas como arestas, conforme proposto por Vultureanu-Albişi et al. (2024).

O grafo de conhecimento é formalizado como um grafo direcionado $G = (V, E)$, onde:

- V é o conjunto de nós, que inclui as entidades de tipos diferentes, como: $V_{\text{habilidade}}$, $V_{\text{educação}}$, $V_{\text{localização}}$, $V_{\text{candidato}}$, $V_{\text{vaga}}$.
- E é o conjunto de arestas, que são as relações entre os nós. Cada aresta $e \in E$ é uma tripla $(u, r, v)$, onde:
  - $u$ é o nó de origem (um tipo de entidade),
  - $v$ é o nó de destino (outro tipo de entidade),
  - $r$ é o tipo de relação entre os nós (por exemplo, tem habilidade, requer habilidade, etc.).

## 4.3 Modelo de Recomendação

No presente trabalho, a recomendação de vagas/candidatos foi modelada como uma tarefa de predição de *links* entre grafos. O modelo, que se trata de uma adaptação do proposto por Vultureanu-Albişi et al. (2024), recebe o grafo $G = (V, E)$ e um conjunto $P$ de possíveis pares de vagas e candidatos, para os quais queremos prever se há uma aresta entre eles. Denotamos esses pares como $(v_i, v_j)$, onde $v_i, v_j \in V$. Para cada $v \in V$, suas *features* são inicializadas com valores aleatórios.

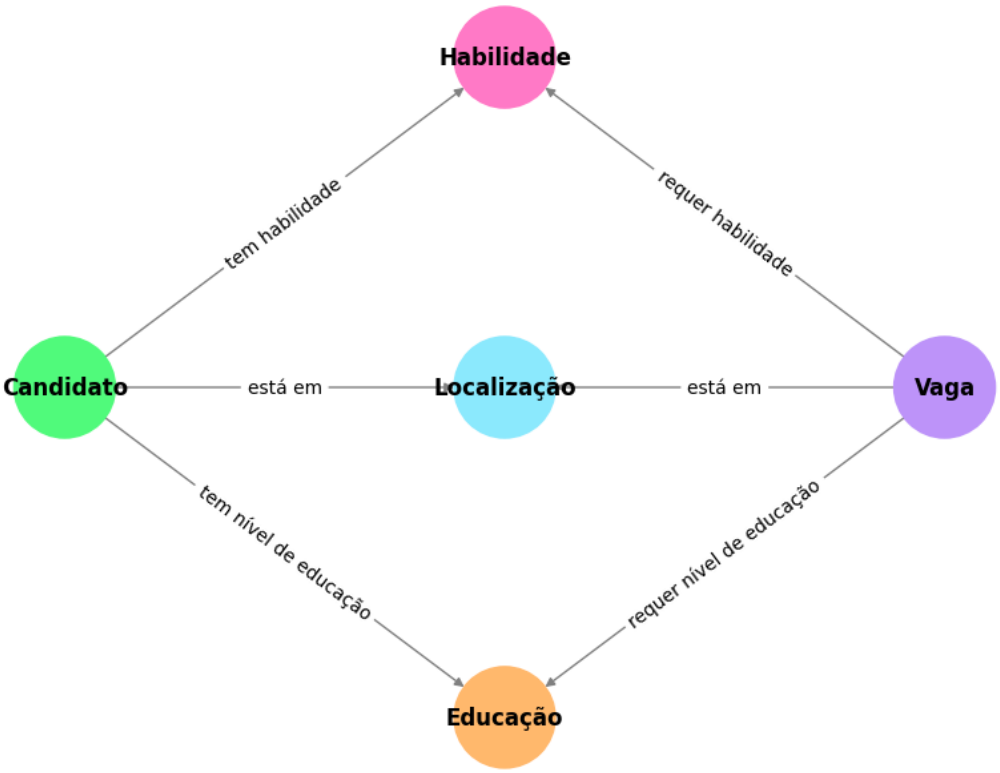


Figura 4.1: Representação do *Dataset* na Forma de um grafo de conhecimento. Fonte: autor.

### 4.3.1 Geração de *Embeddings*

**Camada de Atenção (GATConv)**

Para cada camada de atenção do modelo, a operação básica é dada por:

$$\mathbf{h}_i' = \text{LeakyReLU}\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right)$$

Onde:

- $\mathbf{h}_i'$ é o vetor de características do nó $i$ após a atenção.
- $\mathcal{N}(i)$ é o conjunto de vizinhos do nó $i$.
- $\alpha_{ij}$ é o peso da atenção entre os nós $i$ e $j$, calculado pela fórmula:

$$alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j])\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k])\right)}$$

  Onde $\mathbf{W}$ é uma matriz de pesos para transformação das características dos nós e $\mathbf{a}$ é um vetor de atenção que aprende os pesos entre os pares de nós.

**GATModel**

O modelo de atenção para geração de embeddings (GATModel) é definido como:

$$\mathbf{z} = \text{GATConv}(\text{ReLU}(\text{GATConv}(\mathbf{x}, \mathbf{A})), \mathbf{A})$$

Onde:

- **x** são as *features* dos nós.
- **E** é o conjunto de arestas entre as entidades do grafo.
- ReLU é a função de ativação ReLU.

### 4.3.2 Decodificação e Recomendação

Para cada par $i, j \in P$ a função de decodificação (cálculo de similaridade) é dada por:

$$\hat{y}_{ij} = \text{sim}\left(\mathbf{z}_i, \mathbf{z}_j\right)$$

Onde :

- $\text{sim}(\cdot, \cdot)$ é a função de similaridade cosseno.
- $\mathbf{z}_i$ e $\mathbf{z}_j$ são os embeddings dos nós $i$ e $j$ produzidos pelo GATModel.
- $\hat{y}_{ij}$ é a predição da probabilidade de uma aresta entre os nós $i$ e $j$ existir.

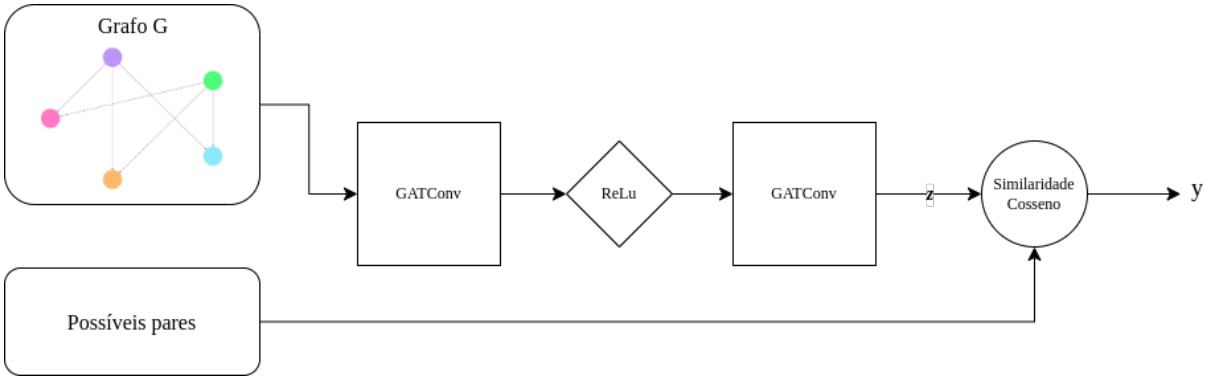


Figura 4.2: Modelo Experimental. Fonte: autor.

### 4.3.3 Otimização e Treinamento

A perda é calculada usando a função de entropia cruzada binária com *logits*, comparando a similaridade predita com o rótulo binário real $y_{ij}$ de cada aresta:

$$\mathcal{L} = -\frac{1}{M} \sum_{(v_i, v_j) \in \mathcal{P}} \left[ y_{ij} \log(\sigma(\text{sim}(v_i, v_j))) + (1 - y_{ij}) \log(1 - \sigma(\text{sim}(v_i, v_j))) \right]$$

onde:

- $P$ é o conjunto de pares de nós possíveis,
- $\sigma(x)$ é a função sigmoide $\sigma(x) = \frac{1}{1+\exp(-x)}$,
- $y_{ij}$ é o rótulo binário real (1 se existe uma aresta entre $v_i$ e $v_j$, 0 caso contrário).

O objetivo é minimizar a perda $\mathcal{L}$ em relação aos parâmetros do modelo $\theta = \{W^{(l)}, a^{(l)}\}$ usando gradiente descendente.

Para avaliar o desempenho e reduzir o impacto de variações nos dados de treino, utiliza-se a técnica de validação cruzada *k-fold*. Nesse processo, o conjunto de dados é dividido em $k$ subconjuntos (*folds*). O modelo é treinado iterativamente em $k-1$ *folds* e avaliado no fold restante, garantindo que cada subconjunto seja utilizado tanto para treino quanto para validação. O resultado final é obtido pela média das métricas calculadas em cada rodada, promovendo uma avaliação mais robusta do modelo.

## 4.4 Análise Estatística

A análise estatística dos resultados do modelo foi realizada em duas etapas principais. Primeiramente, utilizou-se o teste de Shapiro-Wilk[Shapiro and Wilk (1965)] para verificar a normalidade dos dados, considerando a hipótese nula de que os dados seguem uma distribuição normal. Em seguida, como os dados não apresentaram distribuição normal, foi aplicado o teste não paramétrico de Mann-Whitney[Mann and Whitney (1947)] para avaliar diferenças significativas entre os grupos analisados. Esse teste foi escolhido por sua robustez para comparar medianas em amostras independentes sem assumir pressupostos de normalidade.

# 5

# Resultados e Discussões

Este capítulo apresenta os resultados obtidos a partir da implementação e avaliação do modelo experimental, divididos em três seções. A Seção 5.1 descreve as instâncias analisadas no estudo, detalhando os dados utilizados e suas características. A Seção 5.2 discute o ambiente experimental, incluindo as configurações de hardware e software, e as condições em que os experimentos foram realizados. Finalmente, a Seção 5.3 apresenta os resultados experimentais, analisando o desempenho do modelo em relação às métricas propostas, destacando os principais achados e interpretando as implicações dos resultados obtidos.

## 5.1 Instâncias

Para a realização do experimento, foram coletados dados reais registrados na plataforma OxeTech entre os anos de 2021 e 2024. O OxeTech é uma iniciativa do Governo de Alagoas voltada para o desenvolvimento tecnológico do estado. Nela, empresas de tecnologia podem divulgar vagas de emprego e realizar contratações, enquanto pessoas em busca de oportunidade podem se candidatar a essas vagas.

O conjunto de dados utilizado neste estudo contém 281 vagas, 1.250 candidatos, 6.223 candidaturas e 151 contratações. Como detalhado na Seção 4.1, foram extraídas informações sobre habilidades requeridas/possuidas, nível educacional e localização tanto das vagas quanto dos candidatos.

Esses dados foram transformados em uma representação gráfica, resultando em um grafo composto por 2.192 nós, distribuídos entre 281 vagas, 1.250 candidatos, 590 habilidades, 15 níveis educacionais e 56 localizações. Além disso, o grafo contém 21.190 arestas conectando essas entidades.

Os dados de candidaturas e contratações foram utilizados para criar pares candidatos-vaga, os quais servem como exemplos para avaliar o desempenho do modelo, com base no interesse

demonstrado pelas pessoas em busca de emprego e pela decisão de contratação por parte das empresas.

Realizando a amostragem negativa conforme descrito no capítulo 4, o conjunto de pares possíveis (positivos e negativos) para candidaturas totalizou 12.446, enquanto o conjunto de pares para contratações somou 302.

## 5.2 Ambiente

Os experimentos foram realizados utilizando o ambiente fornecido pela plataforma *Kaggle*. O código foi executado em uma instância de GPU T4, com 16 GB de memória, e 1 CPU. A utilização de memória RAM foi limitada a 16 GB, com o espaço em disco configurado para 60 GB.A versão do Python utilizada foi 3.10.12, juntamente com as bibliotecas PyTorch 2.4.1+cu121, torch-geometric 2.6.1, NumPy 1.26.4, pandas 2.1.4 e scikit-learn 1.2.2.

O código, incluindo o ajuste de hiperparâmetros do modelo, está disponível no repositório do projeto para consulta.

## 5.3 Resultados Experimentais

Para avaliar o desempenho do modelo experimental, este foi comparado com um algoritmo recente da literatura, descrito por Vultureanu-Albişi et al. (2024). Como o código-fonte não foi disponibilizado pelos autores, o modelo foi reimplementado com base na descrição contida no artigo original.

Ambos os modelos foram executados 30 vezes para garantir uma amostra estatisticamente significativa para análise. As avaliações consideraram duas perspectivas: o interesse dos candidatos (utilizando dados de candidaturas como amostras positivas e a ausência delas como amostras negativas) e o interesse de contratação das empresas (de maneira análoga, considerando contratações como amostras positivas e sua ausência como negativas). Os resultados foram analisados com base nas métricas apresentadas na Seção 3.4, bem como no tempo de execução de cada modelo. As métricas globais foram calculadas utilizando um limite de 0 na similaridade cosseno entre os nós, considerando como compatível qualquer par com valor superior a 0.

Nas métricas globais (Tabela 5.1), o modelo experimental neste trabalho demonstrou desempenho superior em todas as instâncias de avaliação, com destaque para a predição do interesse dos candidatos (Figura 5.1). Entretanto, na tarefa de predição relacionada às contratações, o modelo apresentou limitações (Figura 5.2). Isso pode ser explicado pela maior escassez de dados nessa tarefa, uma vez que é mais comum que candidatos se inscrevam em várias vagas do que uma vaga tenha múltiplos contratados. Além disso, o conjunto de dados referente às contratações apresentou um volume significativamente reduzido, com apenas 151 exemplos positivos.

| Dataset | Modelo | Acurácia | Precisão | Recall | Tempo de Execução |
|---|---|---|---|---|---|
| Candidaturas | Vultureanu-Albisi | 50.13 | 50.69 | 54.23 | 4481.79 |
| | Modelo Experimental | 76.04* | 74.07* | 81.69* | 1102.77* |
| Contratações | Vultureanu-Albisi | 45.05 | 46.67 | 44.68 | 2646.54 |
| | Modelo Experimental | 51.65* | 53.52* | 54.26* | 1020.33* |

Tabela 5.1: Comparação de Performance Global dos Modelos

Em relação ao ranqueamento (Tabela 5.2), os dois modelos obtiveram desempenho semelhante na predição de candidaturas, com o modelo descrito no artigo original apresentando ligeira vantagem para valores menores de $k$(vide Figura 5.3). No entanto, para a tarefa de predição de contratações, ambos os modelos obtiveram desempenho superior a 94% em todas as métricas avaliadas. Ainda assim, é necessário interpretar esses resultados com cautela, dado o pequeno tamanho e a natureza esparsa do conjunto de dados dessa tarefa.

Além disso, embora tenham sido aplicadas técnicas de amostragem negativa para reduzir a quantidade de dados não informativos, não há garantia de que os exemplos aleatórios selecionados representem adequadamente pares verdadeiramente não similares. Por exemplo, considere um cenário em que a última atividade registrada de um candidato (como sua última candidatura ou atualização de perfil) ocorreu antes da data de criação de uma vaga. Nesse caso, a ausência de interação entre o candidato e a vaga não indica que eles não sejam similares em termos de perfil ou compatibilidade. Em vez disso, reflete apenas a impossibilidade de um "matching"temporal. Incorporar essa distinção no processo de seleção negativa permitiria ao modelo aprender padrões mais precisos e contextualmente relevantes.

Essa abordagem poderia beneficiar o modelo ao evitar falsas suposições de não similaridade, aumentando sua capacidade de identificar combinações realmente não compatíveis. Como resultado, o modelo seria mais robusto e preciso, especialmente em cenários onde a relação temporal influencia a interpretação dos dados.

A inclusão dessas melhorias pode contribuir significativamente para o aumento da acurácia, precisão, e recall globais, bem como para métricas relacionadas ao ranqueamento, como P@k. Incorporar dados temporais permitiria ao modelo identificar padrões associados ao tempo, enquanto exemplos negativos mais representativos aprimorariam sua capacidade de distinguir situações genuinamente dissimilares.

Por fim, em termos de eficiência, o modelo experimental demonstrou-se mais rápido em ambos os conjuntos de dados. Para o conjunto de contratações, o tempo de execução foi reduzido em mais da metade. Já no conjunto de candidaturas, mais volumoso, a redução foi ainda mais expressiva, com o modelo apresentando um tempo de execução quatro vezes menor em comparação ao modelo descrito na literatura.

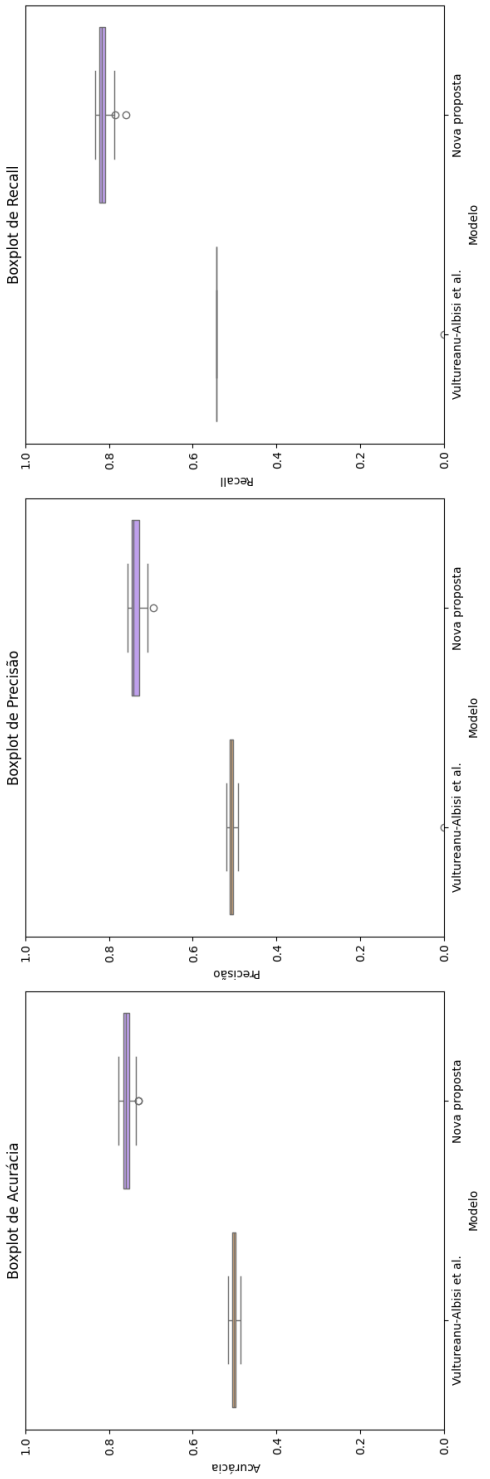


Figura 5.1: Métricas Globais para o *Dataset* de Candidaturas. Fonte: autor.

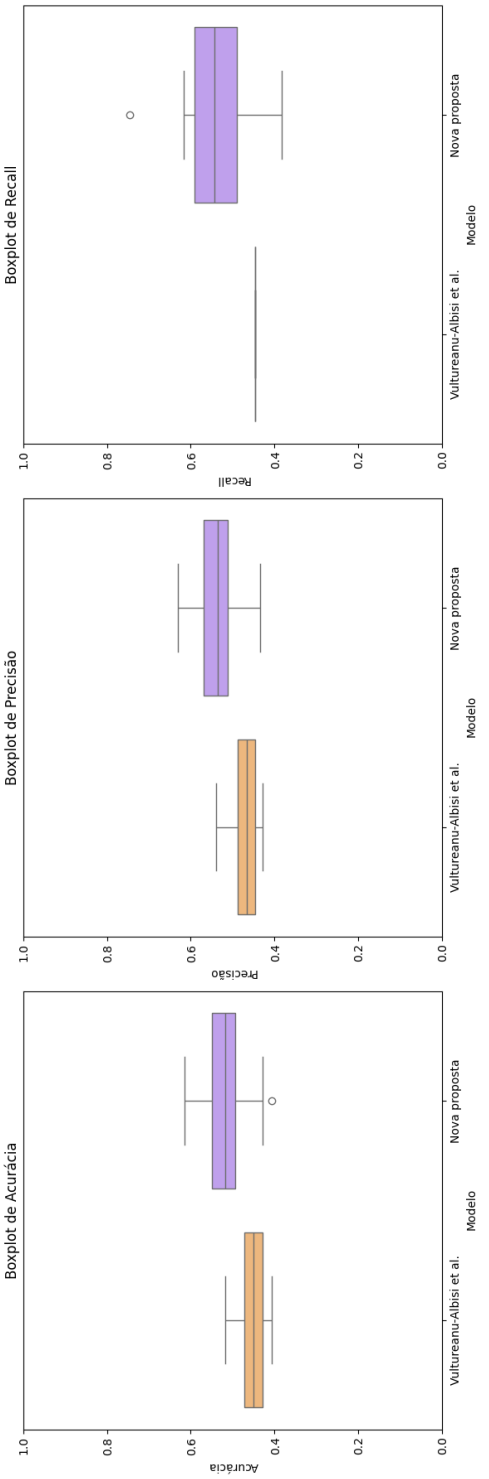


Figura 5.2: Métricas Globais para o *Dataset* de Contratações. Fonte: autor.

| Dataset | Modelo | A@2 | P@2 | R@2 | A@5 | P@5 | R@5 | A@10 | P@10 | R@10 | A@20 | P@20 | R@20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Candidaturas | Vultureanu-Albisi | 90.28 | 68.77 | 89.78 | 99.82 | 66.98 | 89.78 | 100.0 | 66.62 | 97.95 | 100.0 | 66.65 | 99.81 |
| | Modelo Experimental | 95.76* | 77.52* | 92.09* | 100.00 | 69.28* | 92.09* | 100.0 | 67.01 | 98.21* | 100.0 | 66.69 | 99.82* |
| Contratações | Vultureanu-Albisi | 100.00 | 98.4 | 100.00 | 100.00 | 98.4 | 100.00 | 100.00 | 98.4 | 100.0 | 100.0 | 98.4 | 100.0 |
| | Modelo Experimental | 100.00 | 98.4 | 100.00 | 100.00 | 98.4 | 100.00 | 100.00 | 98.4 | 100.0 | 100.0 | 98.4 | 100.0 |

Tabela 5.2: Comparação de Performance de Ranqueamento dos Modelos

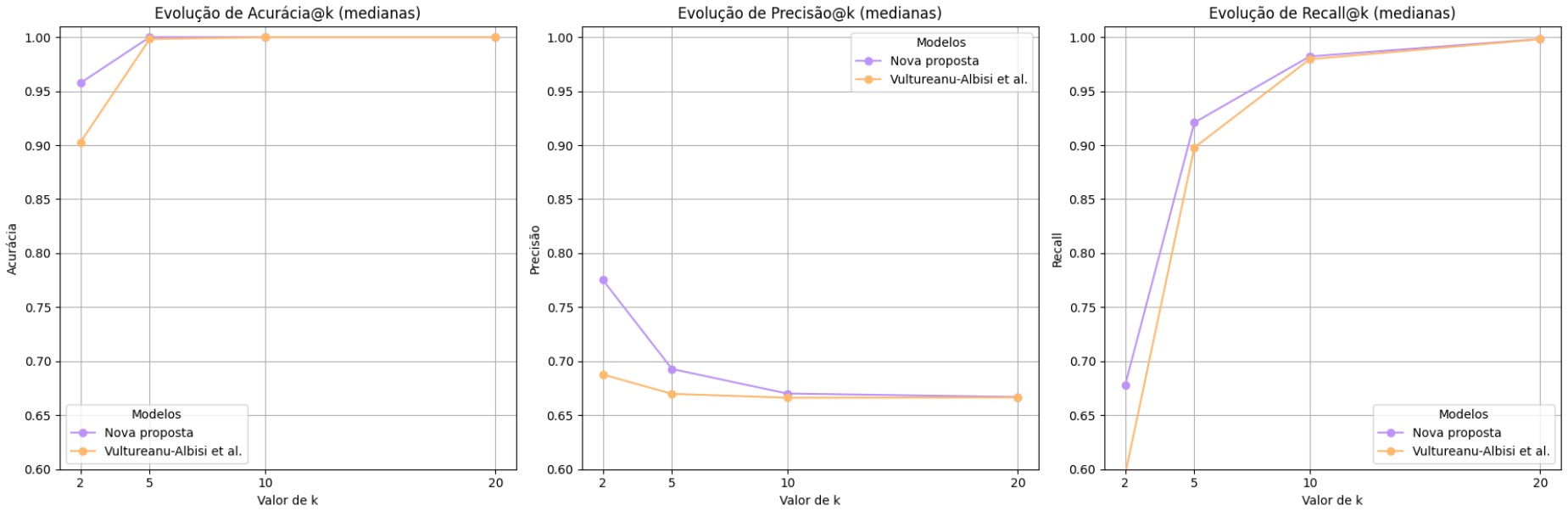


Figura 5.3: Métricas de Ranqueamento para o *Dataset* de Candidaturas. Fonte: autor.

# 6

# Considerações Finais

Este trabalho investigou o ranqueamento e a predição de similaridade entre candidatos e vagas utilizando Aprendizado de Máquina. Para isso, um modelo baseado em *Graph Attention Networks* foi implementado e avaliado em um *dataset* real.

Os resultados demonstraram que o modelo experimental se destacou em termos de métricas globais, particularmente na tarefa de predição de interesse dos candidatos, evidenciando sua capacidade de capturar padrões relevantes neste contexto. Do ponto de vista computacional, o modelo experimental demonstrou maior eficiência, com tempos de execução reduzidos significativamente. No conjunto de dados mais volumoso, referente às candidaturas, a redução foi de até quatro vezes, enquanto no conjunto de contratações, mais enxuto, o tempo de execução foi reduzido pela metade.

Embora promissores, os resultados também revelaram oportunidades de melhoria. A ausência de informações temporais no conjunto de dados limitou a capacidade do modelo de capturar padrões associados à sazonalidade, como diferenças nos fluxos de candidaturas e contratações em períodos de alta ou baixa demanda. Incorporar dados temporais futuros permitiria ao modelo compreender melhor a influência do tempo sobre o comportamento de candidatos e empresas. Além disso, a amostragem negativa aleatória, embora útil para balancear o conjunto de dados, pode ter introduzido ruídos que impactaram a precisão das predições. Métodos mais criteriosos para a seleção de exemplos negativos poderiam aprimorar a capacidade do modelo de distinguir pares verdadeiramente não similares.

Portanto, conclui-se que o modelo experimental representa um avanço significativo em termos de desempenho e eficiência em relação ao estado da arte, com espaço para melhorias que possam torná-lo ainda mais robusto e aplicável a cenários complexos. Espera-se que os *insights* deste trabalho sirvam de base para futuras investigações, contribuindo para o desenvolvimento de sistemas mais precisos e eficientes na área de predição de interesses em processos seletivos.

# Referências bibliográficas


Ali, I., Mughal, N., Khand, Z. H., Ahmed, J., and Mujtaba, G. (2022). Resume classification system using natural language processing and machine learning techniques. *Mehran University Research Journal of Engineering & Technology*, 41(1):65–79.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Governo Federal, g. (2024). Buscar emprego no sistema nacional de emprego (sine). Acesso em: 19 jan. 2025.

Hamilton, W. L. (2020). *Graph representation learning*. Morgan & Claypool Publishers.

Huang, Y., Liu, D.-R., and Lee, S.-J. (2023). Talent recommendation based on attentive deep neural network and implicit relationships of resumes. *Information Processing & Management*, 60(4):103357.

IBGE (2025). Desemprego. https://www.ibge.gov.br/explica/desemprego.php. Acesso em 19 de janeiro de 2025.

Konstantinidis, I., Maragoudakis, M., Magnisalis, I., Berberidis, C., and Peristeras, V. (2022). Knowledge-driven unsupervised skills extraction for graph-based talent matching. In *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*, pages 1–7.

Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019.

Ma, H., Xie, R., Meng, L., Feng, F., Du, X., Sun, X., Kang, Z., and Meng, X. (2024). Negative sampling in recommendation: A survey and future directions. *arXiv preprint arXiv:2409.07237*.

Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

Mao, Y., Lin, S., and Cheng, Y. (2024). A job recommendation model based on a two-layer attention mechanism. *Electronics*, 13(3):485.

Minh, D., Wang, H. X., Li, Y. F., and Nguyen, T. N. (2022). Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, pages 1–66.

Mishra, S., Mallick, P. K., Tripathy, H. K., Jena, L., and Chae, G.-S. (2021). Stacked knn with hard voting predictive approach to assist hiring process in it organizations. *The International Journal of Electrical Engineering & Education*, page 0020720921989015.

Roy, P. K., Chowdhary, S. S., and Bhatia, R. (2020). A machine learning approach for automation of resume recommendation system. *Procedia Computer Science*, 167:2318–2327.

Russell, S. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*.

Saito, Y. and Sugiyama, K. (2022). Job recommendation based on multiple behaviors and explicit preferences. In *2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 1–8. IEEE.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611.

Sugiarto, H. S. (2022). Graph convolution approach for labor market analysis. *Journal of Computational Science*, 64:101855.

Upadhyay, C., Abu-Rasheed, H., Weber, C., and Fathi, M. (2021). Explainable job-posting recommendations using knowledge graphs and named entity recognition. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3291–3296. IEEE.

Vultureanu-Albişi, A., Murareţu, I., and Bădică, C. (2024). A trustworthy and explainable ai recommender system: Job domain case study. In *2024 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–7. IEEE.

Wang, H., Gu, Z., and Gu, M. (2022). Job recommendation algorithm based on knowledge graph. In *2022 IEEE 8th International Conference on Computer and Communications (ICCC)*, pages 2023–2027. IEEE.

Wang, Y., Allouache, Y., and Joubert, C. (2021). A staffing recommender system based on domain-specific knowledge graph. In *2021 Eighth International Conference on Social Network Analysis, Management and Security (SNAMS)*, pages 1–6. IEEE.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Yang, C., Hou, Y., Song, Y., Zhang, T., Wen, J.-R., and Zhao, W. X. (2022). Modeling two-way selection preference for person-job fit. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 102–112.

Yedidia, J. S., Freeman, W. T., Weiss, Y., et al. (2003). Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8(236–239):0018–9448.

Zhou, Z.-H. (2021). *Machine learning*. Springer nature.

Zhu, G., Chen, Y., and Wang, S. (2022). Graph-community-enabled personalized course-job recommendations with cross-domain data integration. *Sustainability*, 14(12):7439.