

Otimização de *Prompts* para Explicações em Sistemas de Recomendação: Balanceando Diversidade e Popularidade

Gabriel Prenassi¹, André Levi Zanon², Henrique Sekido³, Yan Andrade⁴,
Marcelo G. Manzato³, Leonardo Rocha¹

¹Universidade Federal de São João del-Rei, ²Centro Universitário FEI, ³Universidade de São Paulo,

⁴Universidade Federal de Minas Gerais

andrezanon@fei.edu.br; prenassigabriel@aluno.ufsj.edu.br; riqueysekido4@usp.br; yanccomp@dcc.ufmg.br;
mmanzato@icmc.usp.br; lcrocha@ufsj.edu.br

ABSTRACT

Recommender systems are widely used to provide personalized item suggestions, however the increasing complexity of recommendation algorithms has intensified the lack of transparency about why recommendations are generated. Selecting effective explanations remains challenging, especially when selecting attributes (i.e. shared information of different items) that are both diverse and recognizable to users. Popular attributes tend to be more familiar, while diverse attributes help avoid repetitive explanations, creating a trade-off between these two dimensions. This paper investigates the use of prompt optimization for Large Language Models to select explanations in recommender systems that balance attribute diversity and popularity. We use a knowledge graph in the movie domain, connecting items to their attributes, and generate recommendations for users. Then, we apply a prompt optimization strategy to guide the LLM toward explanations that balance the popularity and diversity of attributes. Experiments using literature-based metrics show that optimized prompts can improve the joint balance between these two metrics, outperforming non-optimized prompts and a comparable non-LLM explanation method.

KEYWORDS

Recommender Systems, Explainability, Large Language Models, Prompt Optimization, Knowledge Graphs, Attribute Diversity, Attribute Popularity

1 INTRODUÇÃO

Sistemas de Recomendação (SR) têm como objetivo principal sugerir itens aos usuários com base em seus interesses e interações anteriores [1, 15, 24]. À medida que esses algoritmos se tornaram mais sofisticados para melhorar a qualidade das recomendações, também se tornaram mais opacos, dificultando a compreensão de por que determinado item foi sugerido [12, 18]. Esse cenário torna a explicabilidade particularmente relevante, uma vez que explicações podem contribuir para

aumentar a transparência, confiança, efetividade, persuasão e satisfação do usuário [27].

No contexto de explicações em SR, uma questão central não é apenas como gerar justificativas plausíveis, mas como selecionar informações que sejam úteis do ponto de vista do usuário. A literatura tem mostrado que duas propriedades são particularmente relevantes nesse processo: a diversidade e a popularidade dos atributos utilizados na explicação [3, 28]. A diversidade evita repetição excessiva entre múltiplas explicações, enquanto a popularidade favorece o uso de informações mais familiares e facilmente reconhecíveis. Essas duas dimensões, entretanto, frequentemente se organizam em tensão: privilegiar atributos muito populares tende a reduzir a diversidade, enquanto priorizar diversidade pode levar à seleção de atributos menos reconhecíveis [35]. Assim, a geração de explicações passa a envolver não apenas um problema de linguagem, mas também um de controle de objetivos conflitantes.

Considere, por exemplo, um usuário que assistiu a “Titanic” e recebe como recomendação “Ilha do Medo”. A explicação pode relacionar os filmes por atributos compartilhados, como ator, gênero ou direção. Em listas com múltiplas recomendações, atributos muito frequentes podem tornar as explicações repetitivas, enquanto atributos raros podem prejudicar sua reconhecibilidade. O desafio, portanto, é selecionar explicações que equilibrem diversidade e popularidade.

Modelos de linguagem de grande porte (*Large Language Models* (LLMs)) têm demonstrado potencial para apoiar diferentes tarefas em SR, incluindo reordenação, interação conversacional e geração de explicações [5, 6, 32]. Mais recentemente, técnicas de otimização de *prompt* têm sido exploradas para ajustar iterativamente o comportamento desses modelos, permitindo orientá-los para objetivos específicos [7, 20, 29, 31]. Em recomendação, essa estratégia já mostrou resultados promissores em tarefas como calibração de listas [20], mas permanece pouco explorada no contexto de explicações. Essa lacuna é particularmente relevante porque, nesse cenário, a otimização de *prompt* pode funcionar não apenas como uma técnica para melhorar genericamente a saída textual do modelo, mas como um mecanismo de controle sobre propriedades desejáveis das explicações.

Diante disso, este trabalho investiga se a otimização de *prompts* pode ser utilizada para orientar LLMs na seleção de explicações em SR que equilibrem a diversidade e a popularidade dos atributos. Para isso, adotamos uma abordagem

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia’2026). Lavras/MG, Brazil. Porto Alegre: Brazilian Computer Society, 2026.

© 2026 SBC – Brazilian Computing Society.
ISSN 2966-2753

agnóstica ao recomendador, na qual um Grafo de Conhecimento (GC) no domínio de filmes é utilizado para extrair caminhos explicativos entre itens previamente consumidos e itens recomendados. A LLM, então, atua como seletora desses caminhos, e não como geradora livre de explicações, o que reduz o espaço de saída e torna a avaliação mais controlada. Sobre essa base, aplicamos um processo iterativo de otimização de *prompt* para regular o comportamento do modelo em diferentes objetivos explicativos.

As questões de pesquisa que orientam este trabalho são: (QP1) *Técnicas de otimização de prompt conseguem fazer com que LLMs selecionem explicações que equilibrem diversidade e popularidade de atributos?* e (QP2) *O uso de otimização de prompt gera explicações superiores, em termos de diversidade e popularidade, quando comparado tanto a um prompt não otimizado quanto a um método não baseado em LLMs?*

A principal contribuição deste artigo é mostrar que a otimização de *prompts* pode ser empregada como mecanismo de controle sobre propriedades desejáveis de explicações em SR, e não apenas como uma estratégia genérica para melhorar respostas de LLMs. Para isso, combinamos: (i) um GC no domínio de filmes para gerar caminhos explicativos candidatos; (ii) uma LLM atuando como seletora de explicações; e (iii) um processo iterativo de otimização de *prompt* orientado por métricas de diversidade e popularidade. Em nossos experimentos, a otimização mostrou-se eficaz para direcionar o comportamento do modelo nos objetivos mono-critério e, no cenário multiobjetivo, produziu compensações consistentes entre SEP e ETD, superando *prompts* não otimizados e o baseline ExpLOD em diferentes configurações.

2 TRABALHOS RELACIONADOS

Este trabalho dialoga com três frentes da literatura: explicações em sistemas de recomendação, uso de LLMs para explicabilidade em SR e técnicas de otimização de *prompt*. Em conjunto, essas linhas sustentam o problema investigado neste artigo: como controlar propriedades desejáveis de explicações – em particular, diversidade e popularidade – por meio da otimização do comportamento de LLMs.

2.1 Explicações em Sistemas de Recomendação

Explicações em SR têm sido amplamente estudadas como mecanismo para aumentar transparência, confiança, efetividade, persuasão e satisfação do usuário [25, 27]. Nesse contexto, popularidade e diversidade são propriedades relevantes das explicações [3, 28, 33]: a primeira favorece atributos mais familiares e reconhecíveis, enquanto a segunda evita repetição excessiva ao longo de múltiplas explicações. Como essas propriedades podem entrar em conflito [35], a seleção de explicações passa a envolver um problema de otimização entre objetivos distintos.

2.2 LLMs para Explicabilidade em SR

Trabalhos recentes têm mostrado que LLMs podem apoiar diferentes tarefas em SR, incluindo reordenação de itens, recomendação conversacional e explicabilidade [5, 6, 32]. No caso específico das explicações, Lei et al. [16] demonstram

que LLMs podem reproduzir recomendações e gerar justificativas para modelos caixa-preta, indicando o potencial desses modelos para promover transparência. Em outra direção, Peng et al. [19] combinam representações de usuários e itens à geração textual com GPT-2, mostrando a viabilidade de integrar recomendação e explicação em um mesmo processo otimizado. Apesar desses avanços, a maior parte dessas abordagens concentra-se em gerar explicações plausíveis ou em integrar explicações ao modelo recomendador, sem tratar explicitamente do problema de controlar propriedades específicas da explicação, como diversidade e popularidade. O trabalho mais próximo ao proposto é o de [35], no qual LLMs foram utilizados para balancear essas duas dimensões. No entanto, nesse trabalho, o *prompt* utilizado como entrada para gerar as explicações das recomendações foi definido de forma manual, sem a aplicação de técnicas sistemáticas de otimização de *prompt*. Assim, não foi investigado o impacto da otimização do *prompt* na qualidade das explicações geradas. Para comparar esse trabalho com a nossa proposta, durante nossos experimentos, utilizamos como linha de base o *prompt* inicial, antes da aplicação do processo de otimização.

2.3 Otimização de Prompt em LLMs

A literatura sobre otimização de *prompt* mostra que pequenas mudanças na formulação textual de entrada podem alterar significativamente o comportamento de LLMs [38]. Esse diagnóstico motivou o surgimento de métodos voltados ao refinamento sistemático de instruções. Yang et al. [31], por exemplo, propõem o *Optimization by PROMPTing* (OPRO), no qual a própria LLM gera iterativamente novas versões de *prompts* a partir do histórico de instruções já avaliadas. Wang et al. [29] exploram o *PromptAgent*, aproximando o *prompt* de entrada daquele que seria produzido por um especialista. Zhou et al. [37], por sua vez, investigam a geração e seleção automática de *prompts* pelo próprio modelo. Em conjunto, esses trabalhos mostram que a otimização de *prompt* é uma estratégia promissora para orientar LLMs a objetivos específicos em diferentes domínios [7]. Em recomendação, essa ideia já foi explorada em tarefas como calibração de listas [20], mas permanece pouco investigada no contexto de explicações.

Nosso trabalho se aproxima dessas três frentes ao tratar explicações em SR como objetos que devem exibir propriedades desejáveis, ao explorar LLMs como mecanismo de seleção explicativa e ao empregar otimização de *prompt* para controlar seu comportamento. Ao mesmo tempo, diferencia-se da literatura em um ponto central: em vez de usar LLMs apenas para gerar justificativas ou selecionar explicações de forma genérica, investigamos se a otimização de *prompt* pode funcionar como um mecanismo explícito de controle sobre o *trade-off* entre diversidade e popularidade. Para isso, adotamos uma abordagem agnóstica ao recomendador, baseada em caminhos explicativos extraídos de um GC, e utilizamos a LLM como seletora entre candidatos semanticamente válidos. Com essa estratégia, o artigo avança sobre a literatura ao mostrar que a otimização de *prompt* pode ser empregada não apenas para melhorar saídas textuais em abstrato, mas para

calibrar o comportamento explicativo da LLM sob critérios objetivos e avaliáveis.

3 MÉTODO PROPOSTO

Entre as estratégias para promover a explicabilidade em SR com apoio de LLMs, destacam-se as abordagens agnósticas ao modelo e as intrínsecas. As primeiras geram explicações independentemente dos mecanismos internos do recomendador [12, 16, 35], enquanto as segundas integram recomendação e explicação em um mesmo processo de modelagem [19, 36].

No caso de abordagens agnósticas baseadas em GCs, o objetivo principal é escolher um caminho explicativo capaz de justificar a relação entre o histórico do usuário e o item recomendado. Um caminho de explicação corresponde a uma sequência no GC que conecta um item previamente consumido a um item recomendado por meio de um atributo compartilhado.

Este trabalho propõe uma abordagem agnóstica para seleção de explicações em SRs utilizando LLMs e otimização de *prompts*. A proposta parte de listas de itens previamente recomendados aos usuários e, para cada item recomendado, identifica caminhos explicativos extraídos de um GC no domínio de filmes. Esses caminhos estabelecem conexões entre itens presentes no histórico do usuário e itens recomendados por meio de entidades compartilhadas, como atores, diretores, gêneros ou outras propriedades semânticas.

A LLM atua como seletora de explicações. Para cada item recomendado, os caminhos candidatos são previamente extraídos do grafo e apresentados ao modelo de forma enumerada, cabendo à LLM escolher o caminho que melhor justifica a recomendação. Essa estratégia restringe o espaço de saída a explicações já presentes no GC, reduzindo o risco de alucinação e tornando a avaliação mais controlada. Assim, o algoritmo proposto é organizado em duas etapas principais: (1) **geração dos caminhos de explicação** e (2) **seleção das explicações por meio de LLMs e otimização de *prompts***.

Nesse contexto, investigamos se a otimização automática de *prompts* pode melhorar a seleção dos caminhos explicativos. Para isso, utilizamos o método *OPRO* [31], no qual a própria LLM propõe novas versões de *prompts* a partir do histórico de instruções previamente avaliadas e de seus respectivos desempenhos. As subseções seguintes detalham a geração dos caminhos explicativos, a seleção de explicações com LLMs e o procedimento de otimização via *OPRO* [31].

3.1 Geração dos Caminhos de Explicação

As explicações consideradas neste trabalho são representadas por caminhos em um GC, conectando itens previamente consumidos por um usuário aos itens posteriormente recomendados. Para essa etapa, utilizamos o GC disponibilizado por [34], construído a partir de entidades e relações da Wikidata¹ associadas aos itens do domínio de filmes.

Nesse GC, os filmes são representados como entidades conectadas a atributos semânticos, tais como atores, diretores, gêneros, premiações e demais propriedades disponíveis na

base de conhecimento. As arestas do grafo representam relações entre entidades, permitindo que um item seja conectado a diferentes atributos e que esses atributos sirvam como ponte semântica entre itens distintos.

A partir dessas relações, são gerados caminhos explicativos entre os itens presentes no histórico do usuário e os itens recomendados. Neste trabalho, cada caminho explicativo é composto por três elementos: um item previamente consumido pelo usuário, um atributo intermediário compartilhado e o item recomendado. Dessa forma, um caminho segue a estrutura *item consumido* → *atributo* → *item recomendado*.

Para cada usuário e para cada item recomendado, é construído um conjunto de caminhos candidatos a partir dos atributos compartilhados no GC. Esses caminhos não correspondem necessariamente às explicações finais, mas sim a alternativas possíveis que serão posteriormente avaliadas pelos métodos de seleção de explicações. A etapa descrita nesta subseção é, portanto, responsável por produzir o espaço de candidatos sobre o qual atuam o método proposto e as linhas de base explicadas na Seção 3.2.1.

3.2 Seleção de Explicações e Otimização de *Prompts* com LLMs

A abordagem proposta utiliza a LLM como seletora de caminhos explicativos. Para cada conjunto de recomendações de um usuário, o modelo recebe os caminhos de explicação candidatos gerados na etapa anterior para cada item sugerido e deve escolher aquele que melhor justifica a relação entre o histórico do usuário e o item recomendado. O modelo também recebe informações sobre atributos já utilizados em explicações anteriores do mesmo usuário, permitindo considerar a diversidade das explicações ao longo da lista recomendada.

Em todas as execuções, preservamos a separação entre usuários de treino, validação e teste. O conjunto de treino é utilizado na busca iterativa por instruções que maximizem o objetivo adotado em cada execução, enquanto a validação acompanha esse processo para verificar se o comportamento observado se mantém fora do conjunto diretamente usado na otimização. O conjunto de teste, por sua vez, é reservado à aplicação final da melhor instrução encontrada. Além disso, os itens recomendados e os caminhos candidatos associados a cada recomendação são mantidos fixos ao longo das comparações, de modo que as diferenças observadas reflitam o efeito da instrução utilizada.

3.2.1 Prompt para Escolha das Explicações. O *prompt* passado à LLM para selecionar as explicações é dividido em dois componentes: (1) o *system prompt* e (2) o *user prompt*. O *system prompt* é otimizado pela própria LLM com o algoritmo *OPRO* [31] e define as instruções gerais de comportamento do modelo. Já o *user prompt* reúne os elementos que contextualizam a decisão da LLM e é específico de cada usuário. Devido a restrições de espaço, nesta seção descrevemos como o *prompt* foi construído. Exemplos de uso nos experimentos estão disponíveis no nosso repositório de código-fonte², utilizado para os experimentos aqui reportados.

¹<https://www.wikidata.org/>

²README do repositório no GitHub com exemplos de *prompts*

A formulação do *system prompt* define que a LLM deve retornar apenas os caminhos de explicação escolhidos para cada item recomendado. Essa decisão torna a saída mais estável, reduzindo a chance de alucinações ou de respostas inconsistentes com as opções apresentadas, e facilita o processamento posterior, pois cada resposta pode ser diretamente convertida no caminho explicativo selecionado. Quando a resposta não segue o formato esperado, o sistema realiza uma nova solicitação, reforçando a necessidade de produzir uma escolha válida.

Assim, no *system prompt*, descrevemos inicialmente que: (1) a LLM deve escolher um caminho de explicação para o item recomendado; (2) somente um caminho pode ser escolhido; e (3) a saída deve ser limitada ao espaço de caminhos de explicação gerados pelo procedimento descrito na Seção 3.1 para o item recomendado.

Por fim, de acordo com o objetivo da execução, adicionamos ao *system prompt* uma descrição do critério a ser otimizado: popularidade dos atributos, diversidade dos atributos ou equilíbrio entre ambas. A forma como medimos cada um desses aspectos na etapa de avaliação é apresentada na Seção 4.4.

O *user prompt* reúne os elementos que contextualizam a decisão do modelo. Ele contém: (1) o histórico recente do usuário, que situa a recomendação em seu contexto de consumo; e (2) os caminhos candidatos dos itens do histórico até o item recomendado, delimitando o conjunto de justificativas possíveis.

Como cada *prompt* é utilizado para uma única sugestão, ao explicarmos múltiplas recomendações atribuídas ao mesmo usuário, o *user prompt* também passa a registrar os atributos já utilizados em explicações anteriores.

3.2.2 Processo de Otimização de Prompt. A sensibilidade dos LLMs à formulação do *prompt* é um aspecto já discutido na literatura [38] e também observado em nossos experimentos, como evidenciado pelos resultados apresentados na Seção 5. Por essa razão, adotamos o processo iterativo *OPRO* [31], no qual a própria LLM é utilizada para gerar novas instruções candidatas para o campo *system*. Em outras palavras, o processo reformula progressivamente a instrução que define como o modelo deve se comportar na tarefa de seleção de caminhos explicativos.

Em cada época, a nova instrução gerada pela otimização substitui o componente *system* do *prompt* descrito na Subseção 3.2.1. Em seguida, a seleção das explicações é reaplicada aos mesmos usuários e aos mesmos caminhos candidatos previamente preparados.

Para gerar cada nova instrução do componente *system*, a LLM de otimização recebe um *meta-prompt*. O *meta-prompt* é um *prompt* utilizado para que a LLM gere a nova instrução no *system*. O *meta-prompt* reúne uma nova descrição da tarefa de seleção de caminhos explicativos, restrições fixas de saída, a descrição do critério a ser otimizado na execução e um conjunto reduzido de instruções de referência.

Para definir quais instruções serão efetivamente inseridas no *meta-prompt*, as instruções avaliadas em épocas anteriores são inicialmente ordenadas de acordo com sua pontuação no treino. Em seguida, aplicamos *Maximal Marginal Relevance* (MMR) [10] sobre suas representações textuais para selecionar, dentre elas, aquelas que serão efetivamente exibidas

ao modelo. O uso de MMR permite evitar que o conjunto de referências seja composto por instruções muito repetidas entre si, ainda que bem pontuadas, preservando, ao mesmo tempo, a qualidade do conjunto selecionado. Seja

$$c_{\text{best}} = \arg \max_{c \in C} \text{Score}(c),$$

em que $\text{Score}(c)$ representa a pontuação da instrução c no conjunto de treino. A primeira referência selecionada é c_{best} . Em seguida, a cada passo, o MMR escolhe a próxima instrução candidata $c_i \in C \setminus S$ que maximiza:

$$c^* = \arg \max_{c_i \in C \setminus S} \left[\lambda \text{Sim}(c_i, c_{\text{best}}) - (1 - \lambda) \max_{c_j \in S} \text{Sim}(c_i, c_j) \right],$$

em que C é o conjunto de instruções candidatas avaliadas em épocas anteriores, S é o conjunto de referências já selecionadas e $\text{Sim}(c_i, c_j)$ mede a similaridade entre duas instruções. Em caso de empate na seleção das referências, o desempate é resolvido deterministicamente pela ordem da lista de candidatos, previamente ordenada pelo desempenho no treino. Em nossas execuções, selecionamos três instruções de referência por iteração. Para representar semanticamente os *prompts*, testamos *LLM2Vec* [4] e *SBERT* [21], usando similaridade de cosseno. Também avaliamos $\lambda \in \{0.0, 0.5, 1.0\}$, cobrindo os extremos e um caso intermediário do compromisso entre relevância e diversidade: $\lambda = 0.0$ prioriza diversidade, $\lambda = 1.0$ prioriza similaridade com a melhor instrução em treino e $\lambda = 0.5$ equilibra ambos.

A otimização é conduzida sobre os conjuntos de treino e de validação. Em cada iteração, uma nova instrução é avaliada inicialmente no treino, e seu desempenho orienta a geração das instruções subsequentes. A validação acompanha esse processo para verificar se o comportamento observado no treino também se mantém fora do conjunto diretamente utilizado na busca. A primeira época corresponde à avaliação da instrução original, enquanto as épocas seguintes passam a considerar instruções candidatas geradas a partir das referências selecionadas. Ao final, a instrução de melhor desempenho em treino é armazenada e reaplicada aos usuários de teste. O conjunto de teste também é executado com a instrução original, permitindo comparar diretamente o comportamento do sistema antes e depois da otimização. Em nossas execuções, o processo foi conduzido por até 10 épocas, com validação em todas elas.

Esse procedimento foi repetido para cada conjunto de recomendações utilizado nos experimentos, cujos algoritmos de origem são detalhados na Seção 4.2. Para manter a consistência das comparações, utilizamos a mesma configuração em todas as execuções. Em cada cenário, o modelo recebeu 10 recomendações por usuário e, para cada item recomendado, a LLM foi consultada separadamente para escolher, entre 10 caminhos candidatos associados àquela recomendação, previamente selecionados e reaproveitados ao longo de toda a otimização, um deles como explicação. Dessa forma, as mudanças observadas podem ser atribuídas mais diretamente ao refinamento textual da instrução do que a alterações no contexto apresentado ao modelo. A Figura 1 resume este

fluxo de seleção de explicações e de otimização de *prompts* adotado neste trabalho.

4 AMBIENTE EXPERIMENTAL

4.1 Conjunto de Dados

Os experimentos foram conduzidos utilizando o conjunto de dados *MovieLens ml-latest-small* [13], amplamente utilizado no contexto de Sistemas de Recomendação. Esse conjunto contém 100.836 avaliações realizadas por 610 usuários sobre 9.742 filmes, incluindo identificadores dos usuários e itens, notas atribuídas e *timestamps*. Neste trabalho, o *MovieLens* é utilizado como fonte das interações usuário-item, a partir das quais são treinados os modelos de recomendação responsáveis por gerar os itens posteriormente explicados.

A divisão dos dados foi realizada apenas sobre o conjunto de interações usuário-item do *MovieLens*, utilizado na etapa de **recomendação**. Inicialmente, esse conjunto foi dividido em 90% para desenvolvimento e 10% para teste. Em seguida, o conjunto de desenvolvimento foi novamente dividido em 90% para treinamento e 10% para validação. O treinamento foi utilizado para treinar os modelos de recomendação, enquanto o conjunto de validação foi empregado para a seleção de hiperparâmetros e configurações experimentais. O conjunto de teste foi reservado para a avaliação final dos métodos comparados.

Além das interações do *MovieLens*, utilizamos um GC derivado da *Wikidata*, disponibilizado por [34]. Esse grafo contém entidades do domínio de filmes e relações entre itens e atributos semânticos, como atores, diretores, gêneros e demais propriedades de filmes. O GC é utilizado como fonte das conexões semânticas a partir das quais são gerados os caminhos explicativos candidatos, conforme descrito na Seção 3.1.

A partir das interações dos usuários, das recomendações geradas e do GC, foi construído um conjunto derivado contendo os caminhos explicativos candidatos utilizados pelos métodos avaliados. Esse conjunto associa, para cada usuário e item recomendado, um conjunto de possíveis explicações baseadas em atributos compartilhados entre itens previamente consumidos e itens recomendados. Dessa forma, o *MovieLens* fornece as interações utilizadas pelos recomendadores, enquanto o GC fornece as relações semânticas utilizadas na construção das explicações.

Por fim, para o processo de **otimização** de *prompts*, os usuários presentes no conjunto de caminhos explicativos foram divididos em treinamento, validação e teste, considerando 80% para desenvolvimento e 20% para teste. Do conjunto de desenvolvimento, 80% foi destinado ao treinamento e 20% à validação. Os usuários de treinamento orientaram a otimização iterativa das instruções, os de validação avaliaram sua generalização e os de teste foram reservados para a aplicação final da melhor instrução e para a comparação entre os métodos.

4.2 Geração das Recomendações

A partir da divisão das interações usuário-item do *MovieLens*, descrita na Seção 4.1, foram treinados os modelos de recomendação responsáveis por gerar listas de itens para cada usuário. Essas recomendações servem como ponto de partida para a extração dos caminhos explicativos no GC, uma vez que

cada explicação é construída a partir da relação entre itens previamente consumidos pelo usuário e itens recomendados.

Como o foco deste trabalho está na avaliação de métodos de seleção de explicações, e não na comparação direta entre algoritmos de recomendação, os recomendadores são utilizados como etapa auxiliar para produzir os itens a serem posteriormente explicados. Dessa forma, para cada configuração experimental, os métodos explicativos são avaliados sobre o mesmo conjunto de recomendações e caminhos candidatos. Essa estratégia permite isolar a análise dos métodos de explicação, garantindo que ExpLOD, LLM sem otimização e LLM com otimização via *OPRO* [31] sejam comparados sob as mesmas condições de entrada.

Com o objetivo de avaliar se o comportamento dos métodos explicativos se mantém consistente diante de recomendações produzidas por diferentes famílias algorítmicas, empregamos quatro algoritmos tradicionais de Sistemas de Recomendação. Seguindo as diretrizes de [9], que destacam a importância do rigor metodológico e da reprodutibilidade na avaliação de novas abordagens, selecionamos métodos representativos de diferentes paradigmas. Especificamente, utilizamos User-KNN [23] e Item-KNN [26], representando abordagens baseadas em vizinhança; *Bayesian Personalized Ranking Matrix Factorization* (BPR-MF) [22], como método baseado em fatoração de matrizes; e *Neural Collaborative Filtering* (NCF) [14], como abordagem baseada em redes neurais.

Todos os modelos foram ajustados com o auxílio do *framework Optuna* [2], de modo que cada algoritmo fosse avaliado sob uma configuração otimizada de hiperparâmetros. A seleção dos hiperparâmetros foi realizada com base no desempenho obtido no conjunto de validação, utilizando NDCG@K como métrica de qualidade do ranqueamento. Após o treinamento e a validação dos modelos, avaliamos diferentes tamanhos de lista de recomendação, considerando os valores de K em {1, 5, 10, 20, 50, 100, 200}.

Em uma análise preliminar da qualidade das recomendações, selecionamos $K = 20$ para as etapas subsequentes. Essa escolha foi motivada pelo fato de esse valor apresentar melhor qualidade de ranqueamento, segundo o NDCG@K, permitindo que as explicações fossem geradas a partir de recomendações mais relevantes. Como as métricas de explicabilidade não foram utilizadas para selecionar o valor de K, essa decisão não favorece diretamente nenhum dos métodos explicativos avaliados. O valor de K foi mantido fixo em todos os experimentos posteriores, garantindo que as diferenças observadas nos resultados estejam associadas aos métodos de explicação, e não a variações nas listas de recomendação utilizadas como entrada.

Quanto à implementação, os modelos User-KNN, Item-KNN e BPR-MF foram implementados com a biblioteca *Case Recommender* [8], enquanto o modelo NCF foi implementado com a biblioteca *Microsoft Recommenders* [11]. A partir das listas top-20 geradas por cada algoritmo, foram extraídos os caminhos explicativos no GC.

4.3 Baselines de Explicabilidade

Como linhas de base, utilizamos o (i) ExpLOD [17], um método sintático baseado no ranqueamento de atributos, e

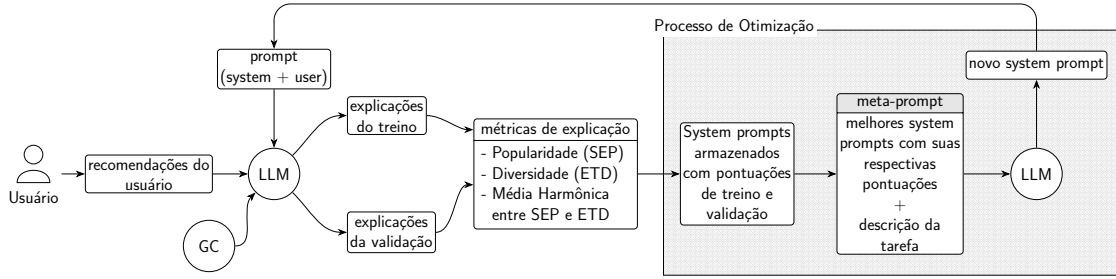


Figura 1: Visão geral da seleção de explicações e da otimização de *prompts*. As explicações de treino e validação são avaliadas pelas métricas, e as melhores instruções em treino compõem o *meta-prompt* da iteração seguinte.

(ii) uma abordagem com LLM sem otimização de *prompt*, em que o modelo escolhe explicações a partir da instrução inicial.

ExpLOD: O algoritmo ExpLOD foi proposto por Musto et al. [17]. O método adapta o TF-IDF para ranquear atributos no grafo bipartido entre itens recomendados e itens previamente interagidos, com o objetivo de gerar explicações, sendo a pontuação definida como $\text{score_explod}(p, I_u, I_r) = \left(\alpha \frac{n_{p, I_u}}{|I_u|} + \beta \frac{n_{p, I_r}}{|I_r|} \right) \cdot \text{IDF}(p)$.

No ExpLOD, cada propriedade que conecta um item interagido a um item recomendado recebe uma pontuação. Após a ordenação dessas pontuações, o caminho explicativo associado ao atributo de maior valor é exibido ao usuário. Para calcular essa pontuação, o número de conexões da propriedade p a partir dos itens interagidos (I_u) é representado por n_{p, I_u} . De forma análoga, n_{p, I_r} representa o número de ligações entre p e o conjunto de itens recomendados (I_r). Essas frequências são então normalizadas e ponderadas pelos parâmetros α e β .

O termo $\text{IDF}(p)$ corresponde ao valor de IDF da propriedade p considerando o conjunto completo de documentos, calculado pela equação $\text{idf}_p = \log \frac{N}{n_{p, N}}$, em que N representa o número total de itens do conjunto de dados e $n_{p, N}$ corresponde ao número de vezes em que um item de N está associado à propriedade p . Neste trabalho, os parâmetros α e β foram definidos como 0,5, conforme especificado no artigo original [17].

LLM sem otimização de *prompt*: Para verificar se a otimização de *prompt* é efetiva, o segundo método utilizado como linha de base corresponde à abordagem com LLM sem otimização. Assim, utilizamos o resultado da primeira época como referência de comparação com a época escolhida após o processo de otimização. Ao utilizar a primeira época, garantimos que o *prompt* base seja comparável àquele selecionado após o processo de otimização, sem introduzir viés adicional na formulação da instrução, uma vez que o *prompt* otimizado é derivado do *prompt* não otimizado.

4.4 Métricas

Para avaliar as explicações geradas, utilizamos duas métricas de explicabilidade propostas por [3]: *Shared Entity Popularity* (SEP) e *Explanation Type Diversity* (ETD). Essas métricas foram escolhidas por capturarem aspectos complementares das explicações geradas a partir do GC. Enquanto o SEP

avalia a popularidade das entidades compartilhadas utilizadas nos caminhos explicativos, o ETD mede a diversidade das propriedades presentes nas explicações selecionadas.

A métrica **Shared Entity Popularity (SEP)** quantifica a popularidade das entidades compartilhadas utilizadas nas explicações. Neste trabalho, assumimos que a popularidade de uma entidade pode ser estimada pelo número de relações em que ela participa no GC. Assim, entidades associadas a um maior número de itens tendem a receber valores mais altos de SEP, indicando que são mais recorrentes e potencialmente mais reconhecíveis no domínio avaliado.

Para cada tipo de entidade, seja $S = [(e_i, v_i)]$ uma lista de entidades ordenada de acordo com sua popularidade, em que e_i representa a i -ésima entidade da lista e v_i representa o número de relações associadas a essa entidade no GC. A popularidade é então calculada de forma suavizada por meio de uma média móvel exponencial, sendo $\text{SEP}(e_i, v_i) = (1 - \beta) \cdot \text{SEP}(e_{i-1}, v_{i-1}) + \beta \cdot v_i$, em que β controla o peso atribuído à popularidade da entidade atual em relação ao valor acumulado das entidades anteriores. Utilizamos $\beta = 0,3$. Após esse cálculo, os valores de SEP são normalizados para o intervalo $[0, 1]$. Dessa forma, valores mais altos indicam explicações baseadas em entidades compartilhadas mais populares no GC, enquanto valores mais baixos indicam entidades menos recorrentes.

A métrica **Explanation Type Diversity (ETD)** mede a diversidade dos tipos de propriedades utilizados nas explicações selecionadas para um usuário. Seja ω_{L_u} o conjunto de propriedades distintas presentes nas explicações associadas às recomendações do usuário u , e ω_L o conjunto total de propriedades disponíveis no GC. Considerando uma lista com k recomendações, o ETD é definido como $\text{ETD} = \frac{|\omega_{L_u}|}{\min(k, |\omega_L|)}$.

Essa métrica assume valores no intervalo $[0, 1]$, em que valores mais altos indicam maior diversidade de propriedades utilizadas nas explicações. Assim, o ETD penaliza métodos que recorrem repetidamente ao mesmo tipo de propriedade explicativa, favorecendo abordagens capazes de selecionar explicações mais variadas ao longo da lista de recomendações.

Além das otimizações orientadas individualmente por SEP e ETD, consideramos também uma configuração de otimização conjunta baseada na média harmônica dessas duas métricas. Essa função-objetivo é utilizada para orientar a busca

por *prompts* que considerem simultaneamente popularidade e diversidade, penalizando soluções muito desbalanceadas entre as duas dimensões. A média harmônica é definida como $MH = 2 \cdot \frac{SEP \cdot ETD}{SEP + ETD}$. Embora essa função seja utilizada como objetivo agregado na otimização, os resultados são reportados separadamente em termos de SEP e ETD. Essa decomposição permite analisar se o comportamento obtido decorre de ganhos em popularidade, ganhos em diversidade ou de diferentes formas de compensação entre ambas.

4.5 Configuração dos Modelos e Reprodutibilidade

Utilizamos a LLM Llama3.1-I (*Meta-Llama-3.1-8B-Instruct*), com quantização em 4 bits, e, para representar os *prompts*, LLM2Vec (*McGill-NLP/LLM2Vec-Meta-Llama-31-8B-Instruct-mntp* e *McGill-NLP/LLM2Vec-Meta-Llama-31-8B-Instruct-mntp-supervised*) e SBERT (*all-MiniLM-L6-v2*). Na seleção dos caminhos, utilizamos temperatura 0,3, top-p 0,9 e até 3 novos tokens, e, na otimização de *prompts*, até 500 novos tokens, temperatura 1,2, top-p 0,9, top-k 40, penalidade de repetição 1,15 e restrição de n-gramas de tamanho 6. Fixamos a semente 2026 na amostragem dos caminhos candidatos, nas execuções com LLM e nas partições de usuários da etapa de explicação, e a semente 42 na busca de hiperparâmetros com *Optuna*. Os experimentos foram executados em Ubuntu 22.04.5 LTS, com AMD Ryzen 9 5900X, 125 GiB de RAM e GPU NVIDIA GeForce RTX 4090 com 24 GiB de memória.

5 RESULTADOS E DISCUSSÃO

Inicialmente, analisamos o comportamento do próprio processo de otimização. Ao todo, foram executados 72 processos de otimização de *prompt*, correspondentes à combinação entre: quatro recomendadores, três métricas-objetivo (SEP, ETD e média harmônica entre SEP e ETD), duas formas de representação dos *prompts* (LLM2Vec e SBERT) e três valores de λ no MMR (0, 0.5 e 1.0). Em 71 desses 72 processos, isto é, em 98.6% dos casos, o melhor *prompt* encontrado ao longo da otimização superou a instrução inicial no conjunto de treino considerando a métrica-objetivo. A implementação dos métodos e das linhas de base, bem como os resultados, está disponível no nosso repositório de código-fonte.³ Dada a quantidade de resultados obtidos, apresentamos nesta seção apenas os casos mais representativos. Ao longo da otimização, a validação foi acompanhada para verificar se o comportamento observado em treino se mantinha em um conjunto não utilizado nesse processo, contribuindo para a análise da generalização das instruções. Para isso, em cada combinação de recomendador e métrica-objetivo, selecionamos, entre as seis configurações avaliadas, aquela com melhor desempenho no treino e reportamos, nas tabelas, seu desempenho no teste.

Portanto, nas tabelas que serão apresentadas, os valores das métricas na linha relativa ao modelo proposto (“LLM otimizado”) representam os resultados de melhor desempenho. Os parâmetros deste são especificados na linha “Configuração” que combina um modelo de representação (LLM2Vec [4] e

SBERT [21]) com um valor de lambda (0, 0.5 e 1) usados no MMR, conforme descritos na Seção 3.2.2. Além disso, realizamos o teste estatístico de Wilcoxon [30] comparando as métricas obtidas por usuário no conjunto de teste pelo modelo proposto com os dois modelos de linha de base (“ExpLOD” e “LLM sem otimização”). O símbolo ▲ significa que o valor do LLM otimizado foi melhor e significativo, já o símbolo ● significa que não houve diferença significativa e o símbolo ▼ significa que o modelo proposto foi pior e significativo.

Por fim, em todos os casos, mesmo quando a otimização foi guiada pela média harmônica entre SEP e ETD, reportamos separadamente essas duas métricas, pois é essa decomposição que permite observar como o equilíbrio é construído, isto é, se ele decorre de ganhos em popularidade, em diversidade, ou de uma compensação entre ambas.

5.1 QP1: Equilíbrio entre diversidade e popularidade

Para responder à QP1, analisamos inicialmente como a otimização de *prompt* afeta a relação entre SEP e ETD, otimizando cada métrica de maneira individual. Assim, a Tabela 1 e Tabela 2 representam a execução do método proposto considerando, respectivamente, apenas a popularidade (SEP) e diversidade (ETD). Nos objetivos mono-critério, o comportamento é claramente antagônico: quando a otimização é orientada a SEP, a popularidade aumenta, mas a diversidade diminui. Quando a otimização é orientada a ETD, ocorre o inverso [35].

Na Tabela 1, a otimização orientada a SEP eleva SEP em todos os recomendadores, enquanto ETD é reduzida em relação ao *prompt* base do LLM. Já na Tabela 2, a otimização orientada a ETD aumenta ETD de forma consistente, mas reduz SEP. Dessa maneira, verificamos que a otimização de *prompt* via LLM consegue otimizar individualmente SEP e ETD. No entanto, otimizar SEP significa que ETD é afetada negativamente e vice-versa. Assim, realizamos um experimento no qual utilizamos otimização de *prompt* utilizando como objetivo a média harmônica entre SEP e ETD.

O resultado deste cenário é apresentado na Tabela 3. Quando passamos ao objetivo conjunto baseado na média harmônica entre SEP e ETD, o comportamento deixa de ser uniforme. Esse é o ponto central da análise de equilíbrio: não esperamos necessariamente melhorar as duas métricas ao mesmo tempo em todos os casos, mas produzir compensações distintas que aumentem o compromisso global entre popularidade e diversidade.

No bloco referente a SEP, User-KNN, BPR-MF e NCF apresentam ganho estatisticamente significativo tanto em relação ao ExpLOD quanto ao *prompt* base. Em Item-KNN, há superioridade em relação ao ExpLOD, mas sem diferença estatística em relação ao *prompt* base. No bloco referente a ETD, apenas Item-KNN apresenta ganho estatisticamente significativo tanto em relação ao ExpLOD quanto ao *prompt* base, enquanto User-KNN, BPR-MF e NCF superam o ExpLOD, mas sem diferença estatística em relação ao *prompt* base. Em termos práticos, isso mostra que o objetivo conjunto não produz o mesmo tipo de ajuste em todos os cenários: embora ainda haja casos em que o equilíbrio se manifesta por

³<https://github.com/andlzanon/prompt-optim-expl-rec>

ganhos mais claros em ETD, o padrão predominante passa a concentrar os avanços mais fortes em SEP.

Assim, em resposta à QP1, os resultados indicam que técnicas de otimização de *prompt* conseguem, sim, fazer com que LLMs selecionem explicações que melhorem individualmente popularidade e diversidade e também equilibrem diversidade e popularidade. No entanto, esse equilíbrio não deve ser interpretado como melhora simultânea e uniforme das duas métricas em todos os casos, mas como um processo de compensação controlada entre elas.

Acreditamos que isso acontece devido à formulação matemática de média harmônica, que não necessariamente busca o equilíbrio entre SEP e ETD, mas sim o cenário em que acontece uma melhora maior em uma métrica sem necessariamente afetar negativamente outra. Nos trabalhos futuros iremos explorar outras formas de agregar SEP e ETD para que ocorra a melhora simultânea.

Tabela 1: Resultados para processos otimizados por SEP.

SEP				
Abordagem	User-KNN	Item-KNN	BPR-MF	NCF
ExpLOD	0.596438	0.560192	0.608610	0.605711
LLM sem otimização	0.626153	0.589186	0.644069	0.608446
LLM otimizado	▲ ▲ 0.685969	▲ ▲ 0.701002	▲ ▲ 0.706125	▲ ▲ 0.730987
Configuração	LLM2Vec $\lambda = 0.0$	LLM2Vec $\lambda = 0.5$	SBERT $\lambda = 0.0$	SBERT $\lambda = 0.5$

ETD				
Abordagem	User-KNN	Item-KNN	BPR-MF	NCF
ExpLOD	0.509016	0.580328	0.524590	0.559016
LLM sem otimização	0.786066	0.797541	0.811475	0.795902
LLM otimizado	▲ ▼ 0.735246	▲ ▼ 0.663934	▲ ▼ 0.706557	▲ ▼ 0.646721
Configuração	LLM2Vec $\lambda = 0.0$	LLM2Vec $\lambda = 0.5$	SBERT $\lambda = 0.0$	SBERT $\lambda = 0.5$

Nota: na linha LLM otimizado, o primeiro símbolo compara com ExpLOD e o segundo com o *prompt* base do LLM, ambos pelo teste de Wilcoxon pareado.

Tabela 2: Resultados para processos otimizados por ETD.

SEP				
Abordagem	User-KNN	Item-KNN	BPR-MF	NCF
ExpLOD	0.596438	0.560192	0.608610	0.605711
LLM sem otimização	0.638907	0.613871	0.649176	0.629463
LLM otimizado	▼ ▼ 0.605605	▼ ▼ 0.526633	▼ ▼ 0.596044	▼ ▼ 0.555479
Configuração	SBERT $\lambda = 1.0$	SBERT $\lambda = 0.0$	SBERT $\lambda = 0.5$	LLM2Vec $\lambda = 1.0$

ETD				
Abordagem	User-KNN	Item-KNN	BPR-MF	NCF
ExpLOD	0.509016	0.580328	0.524590	0.559016
LLM sem otimização	0.772131	0.778689	0.792623	0.767213
LLM otimizado	▲ ▲ 0.827869	▲ ▲ 0.851639	▲ ▲ 0.848361	▲ ▲ 0.841803
Configuração	SBERT $\lambda = 1.0$	SBERT $\lambda = 0.0$	SBERT $\lambda = 0.5$	LLM2Vec $\lambda = 1.0$

Nota: na linha LLM otimizado, o primeiro símbolo compara com ExpLOD e o segundo com o *prompt* base do LLM, ambos pelo teste de Wilcoxon pareado.

5.2 QP2: Comparação com LLM sem otimização e com ExpLOD

Para responder à QP2, analisamos se os *prompts* otimizados escolhem explicações melhores do que aquelas produzidas pelo LLM sem otimização e pelo baseline ExpLOD. Nos objetivos mono-critério, a resposta é direta. Na Tabela 1, quando o alvo é SEP, o *prompt* otimizado supera de forma estatisticamente significativa tanto o ExpLOD quanto o LLM sem otimização em todos os recomendadores. De modo análogo, na Tabela 2, quando o alvo é ETD, o *prompt* otimizado também supera ambos os comparadores em todos os recomendadores.

No objetivo conjunto, a comparação precisa ser interpretada de maneira mais cuidadosa. Na Tabela 3, em SEP, o *prompt* otimizado supera o ExpLOD nos quatro recomendadores e apresenta ganho estatisticamente significativo sobre o

LLM sem otimização em três deles. Em ETD, o *prompt* otimizado também supera o ExpLOD em todos os recomendadores, mas a superioridade em relação ao LLM sem otimização é estatisticamente significativa apenas em Item-KNN. Portanto, nesse cenário, o benefício principal da otimização não é uma dominância simultânea sobre ambos os comparadores em todas as métricas, mas a capacidade de reequilibrar a relação entre diversidade e popularidade.

Assim, em resposta à QP2, os resultados também são positivos. Quando avaliamos a métrica-alvo de cada processo de otimização, os *prompts* otimizados superam de forma consistente o LLM sem otimização e o método ExpLOD nos objetivos mono-critério. No objetivo conjunto, os ganhos são mais seletivos, mas ainda indicam que a otimização melhora o comportamento do modelo em relação ao baseline não baseado em LLMs e, em parte dos casos, também em relação ao próprio LLM sem otimização.

Tabela 3: Resultados para processos otimizados pela média harmônica entre SEP e ETD.

SEP				
Abordagem	User-KNN	Item-KNN	BPR-MF	NCF
ExpLOD	0.596438	0.560192	0.608610	0.605711
LLM sem otimização	0.658011	0.626747	0.662111	0.628054
LLM otimizado	▲ ▲ 0.692547	▲ ▲ 0.621315	▲ ▲ 0.685957	▲ ▲ 0.656289
Configuração	LLM2Vec $\lambda = 0.5$	SBERT $\lambda = 0.0$	SBERT $\lambda = 1.0$	LLM2Vec $\lambda = 0.5$

ETD				
Abordagem	User-KNN	Item-KNN	BPR-MF	NCF
ExpLOD	0.509016	0.580328	0.524590	0.559016
LLM sem otimização	0.757377	0.757377	0.785246	0.776230
LLM otimizado	▲ ▼ 0.748273	▲ ▲ 0.795082	▲ ▼ 0.771311	▲ ▼ 0.752459
Configuração	LLM2Vec $\lambda = 0.5$	SBERT $\lambda = 0.0$	SBERT $\lambda = 1.0$	LLM2Vec $\lambda = 0.5$

Nota: na linha LLM otimizado, o primeiro símbolo compara com ExpLOD e o segundo com o *prompt* base do LLM, ambos pelo teste de Wilcoxon pareado.

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, apresentamos um algoritmo de otimização de *prompt* para seleção de explicações em sistemas de recomendação. Para isso, treinamos diferentes algoritmos de recomendação e extraímos caminhos de explicação de um GC no contexto de filmes. Em seguida, utilizando métricas que quantificam a diversidade e a popularidade das explicações para um usuário, a LLM modifica iterativamente o *prompt* inicial com o objetivo de otimizar ambas as métricas, buscando um equilíbrio entre elas.

Os resultados mostraram que o algoritmo proposto foi bem-sucedido em melhorar cada métrica individualmente. Na perspectiva multiobjetivo, o algoritmo também foi capaz de melhorar as métricas de diversidade e popularidade, sendo que, em mais de metade dos casos, essas melhorias apresentaram significância estatística.

Como limitação, verificamos que a média harmônica talvez não seja a melhor forma de agregar as métricas. Em alguns casos, essa função de agregação pode favorecer uma redução em uma das métricas em troca de um ganho mais expressivo na outra. Como trabalhos futuros, pretendemos investigar outras funções de agregação que favoreçam um melhor equilíbrio entre as métricas. Além disso, planejamos realizar experimentos com usuários para avaliar se as explicações geradas são, de fato, eficazes sob a perspectiva dos usuários.

CUIDADOS ÉTICOS

Utilizamos neste trabalho apenas *datasets* públicos e abertos para fins de pesquisa, já devidamente pseudoanonimizados por seus respectivos criadores originais, o que não permite a reidentificação direta dos usuários. Portanto, esta pesquisa está em conformidade com as diretrizes da Lei Geral de Proteção de Dados Pessoais (LGPD) (Lei n.º 13.709/2018) no que tange ao tratamento de dados para fins acadêmicos. Este estudo não envolveu nova coleta de dados com seres humanos, atendendo, portanto, à Resolução CNS n.º 510/2016.

AGRADECIMENTOS

Esta pesquisa foi financiada por: CNPq, Capes, Fapemig, Fapesp, AWS, NVIDIA, CIIA-Saúde e Instituto Nacional de Ciência e Tecnologia em Inteligência Artificial Responsável para Linguística Computacional, Tratamento e Disseminação de Informação (INCT-TILD-IAR; 408490/2024-1).

Ferramentas de Inteligência Artificial Generativa foram utilizadas apenas como suporte à revisão gramatical de todo o conteúdo. Os autores realizaram uma revisão completa do texto e assumem total responsabilidade pela integridade das informações apresentadas.

REFERÊNCIAS

- [1] Charu C. Aggarwal. 2016. *Recommender Systems*. Springer US. <https://doi.org/10.1007/978-3-319-29659-3>
- [2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. *arXiv:1907.10902 [cs.LG]* <https://arxiv.org/abs/1907.10902>
- [3] Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, and Mirko Marras. 2022. Post Processing Recommender Systems with Knowledge Graphs for Recency, Popularity, and Diversity of Explanations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) (*SIGIR '22*). Association for Computing Machinery, New York, NY, USA, 646–656. <https://doi.org/10.1145/3477495.3532041>
- [4] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dmitriy Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. LLM2Vec: Large Language Models Are Secretly Powerful Text Encoders. In *First Conference on Language Modeling*. <https://openreview.net/forum?id=IW1PR7vEBf>
- [5] Diego Carraro and Derek Bridge. 2025. Enhancing Recommendation Diversity by Re-ranking with Large Language Models. *ACM Trans. Recomm. Syst.* 4, 2, Article 18 (Nov. 2025), 40 pages. <https://doi.org/10.1145/3700604>
- [6] Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, Kai Zheng, Defu Lian, and Enhong Chen. 2024. When large language models meet personalization: perspectives of challenges and opportunities. *World Wide Web* 27, 4 (June 2024), 45 pages. <https://doi.org/10.1007/s11280-024-01276-1>
- [7] Yumin Choi, Jinheon Baek, and Sung Ju Hwang. 2025. System Prompt Optimization with Meta-Learning. In *Advances in Neural Information Processing Systems*, D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (Eds.), Vol. 38. Curran Associates, Inc., 55451–55488. https://proceedings.neurips.cc/paper_files/paper/2025/file/5000f096bed9360a060d835c2a1703bb-Paper-Conference.pdf
- [8] Arthur da Costa, Eduardo Fressato, Fernando Neto, Marcelo Manzato, and Ricardo Campello. 2018. Case recommender: a flexible and extensible python framework for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) (*RecSys '18*). Association for Computing Machinery, New York, NY, USA, 494–495. <https://doi.org/10.1145/3240323.3241611>
- [9] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Trans. Inf. Syst.* 39, 2, Article 20 (Jan. 2021), 49 pages. <https://doi.org/10.1145/3434185>
- [10] Jade Goldstein and Jaime Carbonell. 1998. Summarization: (1) using MMR for diversity - based reranking and (2) evaluating summaries. In *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998* (Baltimore, Maryland) (*TIPSTER '98*). Association for Computational Linguistics, USA, 181–195. <https://doi.org/10.3115/1119089.1119120>
- [11] Scott Graham, Jun-Ki Min, and Tao Wu. 2019. Microsoft recommenders: tools to accelerate developing recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) (*RecSys '19*). Association for Computing Machinery, New York, NY, USA, 542–543. <https://doi.org/10.1145/3298689.3346967>
- [12] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.* 51, 5, Article 93 (Aug. 2018), 42 pages. <https://doi.org/10.1145/3236009>
- [13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. <https://doi.org/10.1145/2827872>
- [14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) (*WWW '17*). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. <https://doi.org/10.1145/3038912.3052569>
- [15] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. 2010. *Recommender systems: an introduction*. Cambridge university press.
- [16] Yuxuan Lei, Jianxun Lian, Jing Yao, Xu Huang, Defu Lian, and Xing Xie. 2024. RecExplainer: Aligning Large Language Models for Explaining Recommendation Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) (*KDD '24*). Association for Computing Machinery, New York, NY, USA, 1530–1541. <https://doi.org/10.1145/3637528.3671802>
- [17] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2016. ExpLOD: A Framework for Explaining Recommendations based on the Linked Open Data Cloud. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) (*RecSys '16*). Association for Computing Machinery, New York, NY, USA, 151–154. <https://doi.org/10.1145/2959100.2959173>
- [18] Ingrid Nunes and Dietmar Jannach. 2017. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction* 27, 3-5 (2017), 393–444. <https://doi.org/10.1007/s11257-017-9195-0>
- [19] Yicui Peng, Hao Chen, Ching-Sheng Lin, Guo Huang, Jinrong Hu, Hui Guo, Bin Kong, Shu Hu, Xi Wu, and Xin Wang. 2024. Uncertainty-Aware Explainable Recommendation with Large Language Models. In *2024 International Joint Conference on Neural Networks (IJCNN)*. 1–8. <https://doi.org/10.1109/IJCNN60899.2024.10651104>
- [20] Gabriel Prenassi, Rodrigo Souza, Henrique Sekido, Guilherme Fonseca, Marcelo Manzato, and Leonardo Rocha. 2025. Calibração de Sistemas de Recomendação com LLMs: Otimização de Prompts para Balancear Precisão, Diversidade e Justiça. In *Proceedings of the 31st Brazilian Symposium on Multimedia and the Web* (Rio de Janeiro/RJ). SBC, Porto Alegre, RS, Brasil, 103–111. <https://doi.org/10.5753/webmedia.2025.15613>
- [21] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. <https://doi.org/10.18653/v1/D19-1410>
- [22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference*

- on *Uncertainty in Artificial Intelligence* (Montreal, Quebec, Canada) (*UAI '09*). AUAI Press, Arlington, Virginia, USA, 452–461.
- [23] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work* (Chapel Hill, North Carolina, USA) (*CSCW '94*). Association for Computing Machinery, New York, NY, USA, 175–186. <https://doi.org/10.1145/192844.192905>
- [24] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. *Introduction to Recommender Systems Handbook*. Springer US, Boston, MA, 1–35. https://doi.org/10.1007/978-0-387-85820-3_1
- [25] Gustavo Santos, Myriam Delgado, Daniel Silver, and Thiago Silva. 2025. Where Next? A Behavioral and Explainable Framework for City and Neighborhood Recommendation. In *Proceedings of the 31st Brazilian Symposium on Multimedia and the Web* (Rio de Janeiro/RJ). SBC, Porto Alegre, RS, Brasil, 482–490. <https://doi.org/10.5753/webmedia.2025.15492>
- [26] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) (*WWW '01*). Association for Computing Machinery, New York, NY, USA, 285–295. <https://doi.org/10.1145/371920.372071>
- [27] Nava Tintarev and Judith Masthoff. 2015. Explaining recommendations: Design and evaluation. In *Recommender systems handbook*. Springer, 353–382.
- [28] Lequn Wang, Jiangwei Pan, Fengdi Che, and Linas Baltrunas. 2026. GenPage: Towards End-to-End Generative Homepage Construction at Netflix. *arXiv preprint arXiv:2606.31031* (2026).
- [29] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. 2024. PromptAgent: Strategic Planning with Language Models Enables Expert-level Prompt Optimization. In *International Conference on Learning Representations*, B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (Eds.), Vol. 2024. 23967–24001. https://proceedings.iclr.cc/paper_files/paper/2024/file/686a3f32067838c8dbb68da6e9e3cf69-Paper-Conference.pdf
- [30] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (1945), 80–83. <http://www.jstor.org/stable/3001968>
- [31] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large Language Models as Optimizers. In *International Conference on Learning Representations*, B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (Eds.), Vol. 2024. 12028–12068. https://proceedings.iclr.cc/paper_files/paper/2024/file/3339f19c5fcee3ad74502947a32be9e6-Paper-Conference.pdf
- [32] Dayu Yang, Fumian Chen, and Hui Fang. 2024. Behavior Alignment: A New Perspective of Evaluating LLM-based Conversational Recommendation Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) (*SIGIR '24*). Association for Computing Machinery, New York, NY, USA, 2286–2290. <https://doi.org/10.1145/3626772.3657924>
- [33] André Zanon, Leonardo Rocha, and Marcelo Manzato. 2024. O Impacto de Estratégias de Embeddings de Grafos na Explicabilidade de Sistemas de Recomendação. In *Proceedings of the 30th Brazilian Symposium on Multimedia and the Web* (Juiz de Fora/MG). SBC, Porto Alegre, RS, Brasil, 231–239. <https://doi.org/10.5753/webmedia.2024.241857>
- [34] André Levi Zanon, Leonardo Chaves Dutra da Rocha, and Marcelo Garcia Manzato. 2024. Model-agnostic knowledge graph embedding explanations for recommender systems. In *World Conference on Explainable Artificial Intelligence*. Springer, 3–27.
- [35] André Levi Zanon, Leonardo Chaves Dutra da Rocha, and Marcelo Garcia Manzato. 2025. Multi-Objective Optimization of Explanation Metrics in Recommender Systems with LLMs. In *2025 IEEE 37th International Conference on Tools with Artificial Intelligence (ICTAI)*. 149–156. <https://doi.org/10.1109/ICTAI66417.2025.00027>
- [36] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101. <https://doi.org/10.1561/15000000066>
- [37] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitit, Harris Chan, and Jimmy Ba. 2023. Large Language Models are Human-Level Prompt Engineers. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=92gvk82DE->
- [38] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, and Xing Xie. 2024. PromptRobust: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis* (Salt Lake City, UT, USA) (*LAMPS '24*). Association for Computing Machinery, New York, NY, USA, 57–68. <https://doi.org/10.1145/3689217.3690621>