O Impacto de Estratégias de *Embeddings* de Grafos na Explicabilidade de Sistemas de Recomendação

André Levi Zanon
andrezanon@usp.br
Universidade de São Paulo
São Carlos, São Paulo, Brasil

Leonardo Rocha
lcrocha@ufsj.edu.br
Universidade Federal de São João del-Rei
São João del Rei, Minas Gerais, Brasil

Marcelo Garcia Manzato
mmanzato@icmc.usp.br
Universidade de São Paulo
São Carlos, São Paulo, Brasil

ABSTRACT

Explanations in recommender systems are essential in improving trust, transparency, and persuasion. Recently, using Knowledge Graphs (KG) to generate explanations gained attention due to the semantic representation of information in which items and their attributes are represented as nodes, connected by edges, representing connections among them. Model-agnostic KG explainable algorithms can be based on syntactic approaches or graph embeddings. The impact of graph embedding strategies in generating meaningful explanations still needs to be studied in the literature. To fill this gap, in this work, we evaluate the quality of explanations provided by different graph embeddings and compare them with traditional syntactic strategies. The quality of explanations was assessed using three metrics from the literature: diversity, popularity and recency. Results indicate that the embedding algorithm chosen impacts the quality of explanations and generates more balanced results regarding popularity and explanation diversity compared to syntactic approaches.

KEYWORDS

Sistemas de Recomendação, Explicações, *Embedding* de Grafos

1 INTRODUÇÃO

Sistemas de Recomendação (SsR) são algoritmos que oferecem sugestões a usuários não apenas com base em seus históricos de interações, mas também em interações de outros usuários, metadados de itens, conhecimento de domínio e informações contextuais [25]. Para usufruir de todos essas informações, as arquiteturas de SsR vêm se apresentando cada vez mais complexas e, consequentemente, pouco transparentes quanto às explicações das recomendações realizadas aos usuários, tornando-se verdadeiras caixas pretas [31]. No entanto, gerar explicações para recomendações têm recebido uma significativa atenção da literatura [3, 22], pois podem melhorar os SsR ao fornecer transparência, confiança, eficácia, persuasão, e satisfação [31].

Atualmente, as estratégias para geração de explicações para SsR podem ser categorizadas, basicamente, em duas abordagens: 1) intrínsecas ao modelo; e 2) agnósticas ao modelo, chamadas de pós-hoc [26, 37]. Modelos intrínsecos buscam gerar explicações em conjunto com a recomendação em si, apresentando os motivos pelos quais um item interagido está relacionado com o recomendado [33]. Métodos agnósticos ou pós-hoc, por outro lado, utilizam um algoritmo desassociado da recomendação para relacionar itens interagidos com itens recomendados e, portanto, não dependem do algoritmo de recomendação [22]. Métodos agnósticos, geralmente, são enriquecidos por fontes externas, como Grafos de Conhecimento (GCs), em que nós representam itens e atributos relacionados a estes e arestas representam as relações semânticas entre os nós [5], sendo esse o foco do presente trabalho.

Os atuais algoritmos GC pós-hoc predominantemente utilizam abordagens sintáticas em que a relevância dos caminhos que conectam itens interagidos e recomendados é medida por meio do número de links associados. No entanto, esta medida pode não capturar completamente as relações semânticas entre os dados. Em contrapartida, abordagens baseadas em *embedding* de grafos [2, 17] podem gerar representações semânticas de caminhos entre nós de itens recomendados e interagidos ao projetá-los em um espaço vetorial. Apesar do potencial impacto que a escolha do algoritmo de *embedding* de grafo utilizado pode ter na qualidade das explicações das recomendações realizadas [21], não encontramos na literatura trabalhos que realizam essa avaliação.

Assim, o objetivo principal desse artigo visa preencher essa lacuna da literatura e comparar o impacto de diferentes algoritmos de *embedding* de grafos na qualidade de explicação em algoritmos de pós-hoc de explicação em sistemas de recomendação, comparando-os também com abordagens sintáticas. Portanto, os objetivos específicos foram divididos em duas questões de pesquisa. A primeira é: **QP1: Qual o impacto de diferentes tipos de algoritmos de *embedding* de grafos na qualidade das explicações geradas para SsR?** A fim de responder a essa pergunta de pesquisa, implementamos um algoritmo agnóstico a modelo (pós-hoc) de explicação utilizando três modelos de *embeddings* de grafos: um bilinear e dois translacionais. Estas representações vetoriais dos nós e arestas dos grafos foram geradas por três diferentes estratégias consideradas estado-da-arte (i.e. TransE [18] e RotatE [29] translacionais e ComplEx [32] bilinear). Os *embeddings* obtidos por esses algoritmos foram combinados para gerar representações do usuário e dos caminhos do grafo que conectam itens interagidos e recomendados, sendo que, o caminho com representação em espaço latente mais similar à representação do usuário é escolhido como explicação ao usuário. Nas análises experimentais, consideramos métricas que avaliam a qualidade da explicação medindo a atualidade dos itens interagidos e a popularidade e diversidade dos atributos que conectam os itens interagidos ao item recomendado [2].

De forma complementar, comparamos as estratégias de *embeddings* de GCs com três abordagens sintáticas (ExpLOD [19], ExpLOD

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2024). Juiz de Fora, Brazil. 2024.

ISSN 2966-2753

versão 2 (ExpLOD v2) [20] e o *Proposed Property-based Explanation Model (PEM)* [10]). O objetivo dessa análise é responder nossa segunda pergunta de pesquisa: **QP2: As estratégias baseadas em *embeddings* de grafos são, de fato, capazes de gerar explicações melhores em SsR em comparação às abordagens sintáticas?**.

Portanto, as principais contribuições deste trabalho são:

- Análise comparativa entre estratégias de *embedding* de grafos e algoritmos sintáticos na qualidade das explicações geradas em SsR;
- Análise comparativa do impacto de diferentes estratégias de *embedding* de grafos na qualidade das explicações geradas para SsR;
- Criação de um arcabouço completo para avaliação de estratégias de geração de explicações para SsR baseadas em grafos de conhecimento.

Focando primeiramente na QP1, nossos resultados deixam claro que modelos bilineares, capazes de representar relações mais complexas entre nós e arestas, impactaram positivamente nas métricas de qualidade de explicação. Com relação à QP2, observamos que enquanto métodos sintáticos priorizam a recência dos itens e popularidade de atributos escolhidos nas explicações, estratégias de *embeddings* conseguem balancear o *trade-off* entre a popularidade e diversidade de atributos de itens nas explicações mostradas aos usuários.

O artigo está estruturado da seguinte forma: A Seção 2 revisa trabalhos relacionados a algoritmos de SsR explicáveis e baseados em GCs. A Seção 3 detalha o arcabouço de avaliação focado em reprodutibilidade, incluindo a configuração experimental, métricas, conjuntos de dados e algoritmos. A Seção 4 discute os resultados, e a Seção 5 resume as descobertas.

2 TRABALHOS RELACIONADOS

Uma vez que GC fornecem metadados estruturados sobre itens, eles têm sido utilizados para gerar recomendações precisas e explicáveis em diversas arquiteturas de recomendação. Explicações em GC são geradas a partir da associação entre itens interagidos e recomendados com atributos compartilhados. Na literatura existem duas abordagens de algoritmos explicativos pós-hoc ou agnósticas ao modelo com CG: uma em que as recomendações são reordenadas com base nas melhores explicações para um item recomendado e outra em que somente as explicações são geradas [22].

Considerando algoritmos de reordenação agnósticos ao modelo utilizando GC, [2] utilizou três métricas de otimização - atualidade de itens interagidos, popularidade e diversidade de atributos extraídos de caminhos de explicação do GC - para reordenar recomendações. Por sua vez, [34] reordenou recomendações avaliando a relevância dos atributos extraídos de caminhos de explicação ao comparar a frequência de associações de atributos com itens interagidos e com o catálogo de itens. Além disso, [14] gerou explicações por meio de extração de aspectos e análise de sentimento, aumentando a precisão da recomendação ao incorporar avaliações textuais como regularizador de algoritmo de recomendação. No entanto, nesses trabalhos, as abordagens propostas avaliaram explicações com base exclusivamente em métricas como precisão e diversidade.

Considerando arquiteturas GC pós-hoc para gerar explicações, [19] criou um algoritmo chamado ExpLOD, que alavanca explicações de um GC com base em um grafo bipartido que conecta itens interagidos com recomendados pelos atributos que compartilham. As explicações são classificadas com base em uma adaptação da métrica *Term-Frequency Inverse Document Frequency* (TF-IDF), onde os nós de item são documentos e os nós de atributo são termos. Este trabalho foi avaliado comparando a explicação proposta com informações extraídas do GC em um experimento online, onde o método alcançou percepção do usuário melhorada considerando objetivos de explicação. [20] estendeu o ExpLOD [19], adicionando atributos mais amplos da hierarquia GC. O mesmo experimento online foi conduzido, mas as explicações do ExpLOD foram comparadas com a nova versão proposta. Os usuários preferiram explicações com atributos mais amplos ao analisar sob a perspectiva de objetivos de explicação. Mais recentemente, [10] propôs o Modelo de Explicação Baseado em Propriedades (PEM), uma função de pontuação que classifica atributos com base em suas conexões com itens interagidos e o catálogo completo de itens. PEM superou a segunda versão do ExpLOD em experimentos online, estabelecendo-se como o estado-da-arte para algoritmos explicativos de GC agnósticos ao modelo. Entretanto, essas abordagens são sintáticas e não consideram intrinsecamente a estrutura e o caminho do GC para gerar explicações. Para suprir essa lacuna, em [35] foi criado um algoritmo de explicação agnóstico ao sistema de recomendação utilizando representações vetoriais de GC e comparado com abordagens sintáticas utilizando métricas offline de qualidade de explicação. No entanto, diferentes maneiras de gerar representações vetoriais de grafos não foram exploradas visto que somente um modelo para a geração dos *embeddings* de grafos foi utilizado no algoritmo de explicação.

A avaliação de explicações também tem recebido atenção, já que as explicações são principalmente avaliadas com base em testes de usuários online, que são demorados e custosos para validar as estratégias. Em [6], métricas offline foram implementadas para avaliar recomendações explicáveis, no entanto, avaliando a robustez dos algoritmos explicativos medindo o número de itens que podem ser explicáveis para os usuários e o número de interações do usuário relacionadas às explicações. Por outro lado, alguns trabalhos também consideram a diversidade e relevância dos atributos exibidos nas explicações [2, 27] embora a relação entre tais métricas e testes online seja definida. Além disso, métricas offline não são padronizadas, e trabalhos que avaliam com estudos de usuários carecem de avaliação quantitativa.

Assim, SsR explicáveis frequentemente não avaliam as explicações de forma quantitativa e qualitativa, já que modelos agnósticos de reordenação com GC contribuem com métricas de precisão e diversidade. Por outro lado, explicações com GC agnósticas ao modelo são avaliadas com testes de usuário online, que são custosos e limitados à avaliação do número de participantes e, como resultado, não são extensivamente avaliados de maneira offline.

3 MATERIAIS E MÉTODOS

3.1 Visão Geral

A fim de responder a QP1 e analisar como diferentes algoritmos de *embeddings* de grafos impactam na geração de explicações agnósticas a modelo em SsR, apresentamos um arcabouço que gera

explicações para recomendações a partir de *embeddings* de grafos gerados pelos algoritmos TransE [18], ComplEx [32] e RotatE [29]. O caminho mostrado como a explicação escolhida é dado pela maior similaridade entre dois *embeddings*: o do caminho e o do usuário. O *embedding* do caminho é composto pela soma dos *embeddings* dos nós e arestas que conectam um ou mais nós de itens interagidos pelo usuário com um nó de item recomendado. Já o *embedding* do usuário é composto pela somatória dos *embeddings* dos nós dos itens interagidos. Na Seção 3.2 detalhamos as estratégias baseadas em *embeddings*.

Para responder a QP2, comparamos as abordagens de *embedding* de grafos com três algoritmos de estado-da-arte para explicações sintáticas em SsR. Nosso objetivo é verificar se abordagens baseadas em *embeddings* performam melhor que abordagens sintáticas. Os algoritmos sintáticos implementados foram: ExpLOD [19], ExpLOD versão 2 (ExpLOD v2) [20] e o *Proposed Property-based Explanation Model (PEM)* [10]. Os três utilizam estratégias para balancear a quantidade de referências que nós de atributos possuem entre itens interagidos e recomendados para escolher o caminho mais relevante para uma explicação. Na Seção 3.3 detalhamos essas abordagens. Na Seção 3.4 detalhamos as métricas propostas por [2] que medem a diversidade de atributos, a popularidade dos mesmos e a recência de itens dentro das explicações para avaliar a qualidade das explicações.

As estratégias avaliadas são pós-hoc e, portanto, independem dos SsR. Em nossa avaliação consideramos sete algoritmos de recomendação baseados em diferentes abordagens: Mais Popular [7] para recomendações não personalizadas, o algoritmo PageRank Personalizado [19] aumentado com o grafo Wikidata para recomendações baseadas em grafo, User-KNN [24] para algoritmos de vizinhança, *Embarrassingly Shallow AutoEncoder* (EASE) [28] e *Bayesian Personalized Ranking Matrix Factorization* (BPR-MF) [23] para algoritmos não neurais, e *Neural Collaborative Filtering* (NCF) [16] para arquiteturas baseadas em redes neurais. Utilizamos a biblioteca CaseRecommender [8] para implementar os algoritmos Mais Popular, User-KNN e BPR-MF. As implementações dos outros recomendadores, juntamente com os algoritmos de explicação, métricas e consultas para extrair as triplas para a construção do GC, estão **disponíveis em um repositório de código aberto[1], sendo essa uma de nossas contribuições nesse trabalho**.

Nas avaliações, consideramos os conjuntos de dados MovieLens 100k [15] e da LastFM [4] para as explicações das Top-5 recomendações de seis algoritmos de SsR para todos os usuários. Assim, para os itens mais bem ranqueados de cada algoritmo de recomendação, todos os sete (quatro de *embedding* e três sintáticos) algoritmos de explicação agnósticos ao modelo foram executados a fim de obter as métricas de qualidade de explicação. Considerando as orientações de reprodutibilidade em SsR propostas [12] e robustez da avaliação de explicação destacados [30], e para garantir uma avaliação rigorosa, aplicamos seis algoritmos de SsR de diferentes famílias, utilizando 90% do conjunto de dados para treinamento e 10% para teste para gerar as explicações.

Utilizamos um GC da Wikidata *Linked Open Data* (LOD) para os domínios de filmes e artistas musicais para implementar todos os algoritmos avaliados. Os dados processados permaneceram com 99% das interações originais para o conjunto de dados MovieLens 100k e 89% para o conjunto de dados LastFM. Um resumo das estatísticas do MovieLens 100k e do LastFM antes e depois do pré-processamento e as informações do GC estão disponíveis na Tabela 1.

		MovieLens	LastFM
Conjunto Original	usuários	610	1,892
	itens	9,724	17,632
	interações	100,836	92,834
Conjunto Processado	usuários	610	1,875
	itens	9,517	11,641
	interações	100,521	83,017
Wikidata GC	entidades	78,703	34,297
	triplas	295,787	134,197
	tipos de arestas	23	33

Tabela 1: Estatísticas dos conjuntos de dados originais e processados, e informações do GC quanto ao número de entidades, triplas e quantidade de arestas.

3.2 Abordagens Baseadas em *Embedding*

Explicações em GC agnósticas a modelo tem por objetivo descobrir qual o caminho mais relevante que conecta um item interagido a um recomendado. A equação 1 define formalmente esse objetivo em que para todos os caminhos c no conjunto de caminhos C que conectam um item interagido a um recomendado, uma função de agregação (agg) - como média e soma - é aplicada sob a relevância rel de cada nó n em c.

$$argmax(\forall c \in C : agg(rel(n) \forall n \in c) \tag{1}$$

A Figura 1 ilustra a abordagem de *embeddings* de GC agnóstica ao modelo. Os nós interagidos pelo usuário no GC são os azuis, os nós de atributo são representados em amarelo, e o nó em vermelho é o do item recomendado. As mesmas cores se aplicam aos vetores que representam os *embeddings* desses nós. O vetor preto representa um *embedding* de relações entre nós, representadas por uma seta de mesma cor na Figura 1.

Dois *embeddings* são necessários para calcular a relevância rel de um caminho de explicação: o do usuário e o do caminho. O *embedding* do usuário, que é calculado por um *pooling* de soma dos *embeddings* dos itens interagidos; e o *embedding* do caminho, que, por sua vez, é um *pooling* de soma de todos os *embeddings* de itens, atributos e relações do caminho que conecta um nó de item interagido pelo usuário a um nó de item recomendado no GC. As Equações 2 e 3 exibem o cálculo para cada um dos *embeddings* onde I é o conjunto de nós de itens interagidos pelo usuário, e P é o conjunto de nós de itens, relações e atributos em um caminho. O método *embedding* retorna o *embedding* do nó passado como parâmetro. Os caminhos foram extraídos usando o algoritmo de Dijkstra [9].

$$embed(user) = \sum_{i \in I} embedding(i) \tag{2}$$

$$embed(path) = \sum_{n \in P} embedding(n) \tag{3}$$

[1] https://github.com/andlzanon/lod-personalized-recommender

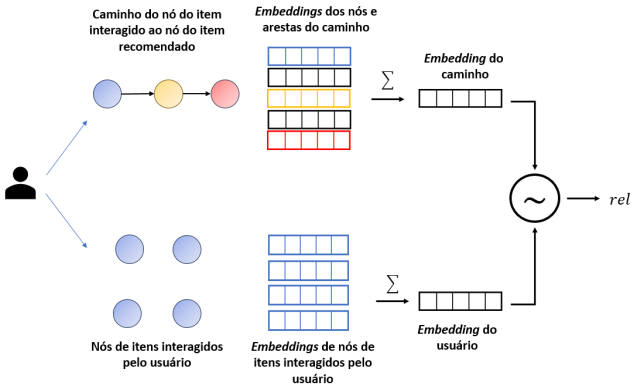

Figura 1: Estrutura do modelo proposto. Os nós azuis representam os itens interagidos pelo usuário, os nós amarelos representam nós de atributo, e o nó vermelho é o nó do item recomendado. O mesmo esquema de cores se aplica aos vetores gerados pelos algoritmos de *embeddings*. Vetores pretos representam o *embeddings* de arestas. O símbolo $\sum$ representa uma operação de *pooling* de soma, e $\sim$ é a similaridade de cosseno entre dois *embeddings*, rel é a saída da função de similaridade de cosseno.

O caminho de explicação escolhido é aquele com a maior similaridade com o *embedding* do usuário entre todos os caminhos que conectam pelo menos um item interagido ao item recomendado. Nesse sentido, a similaridade de cosseno do *embedding* do usuário com os *embeddings* do caminho computa essa proximidade. O comprimento máximo do caminho foi definido como 5, e o número de itens interagidos por explicação, de 3, seguindo a mesma configuração dos *baselines*. O valor da similaridade é representado na Equação 4, onde $embed(user)$ é o *embedding* do usuário e $embed(path)$ é o *embedding* do caminho.

$$sim(user, path) = \frac{embed(user).embed(path)}{||embed(user)||.||embed(path)||} \quad (4)$$

A seleção do caminho de explicação respeita a Equação 5, em que, para todos os caminhos de explicação dentro do conjunto $Paths$ que ligam um nó de item interagido a um nó de item recomendado por atributos compartilhados entre eles, o *embedding* do caminho com a maior similaridade de cosseno com o *embedding* do usuário é o escolhido.

$$argmax(\forall path \in Paths\ sim(user, path)) \quad (5)$$

O *Algorithm 1* é o pseudocódigo do algoritmo de explicação baseado em *embedding* e utiliza de três parâmetros: os itens de histórico do usuário ($profile\_items$), o item recomendado a ser explicado (rec_item) e o modelo de *embbedding* de grafos treinado ($embed\_model$). Na linha 2 o *embbedding* do usuário é criado a partir da soma dos *embbedding* dos nós dos itens interagidos e que são retornados pelo modelo de *embbedding* de grafos.

Em seguida, na linha 3, todos os caminhos de um item interagido ao recomendado são obtidos utilizando o algoritmo de Dijkstra [9]. Como no algoritmo proposto o *embbedding* do usuário é comparado aos *embbeddings* dos caminhos, as linhas 4 e 5 inicializam variáveis que serão responsáveis por armazenar a similaridade máxima atual e o caminho de *embbedding* mais similar ao do usuário.

As linhas 6 a 14 representam a geração dos *embbeddings* dos caminhos e a comparação com o *embbedding* do usuário. Dessa forma, para cada caminho, o *embbedding* é gerado somando os embeddings dos nós que o compõem. Portanto, $c.nos()$ retorna uma lista de nós do caminho c e $embed\_model(c.nos())$ retorna os *embbeddings* desses nós. Em seguida, a similaridade de cosseno entre os *embbeddings* do usuário e do caminho é realizada na linha 8. Quando esse valor é maior que o máximo, tanto o valor máximo, quanto o caminho que corresponde a essa similaridade máxima, são atualizados nas linhas 10 e 11. Na linha 14 o algoritmo retorna o caminho de *embbedding* com maior similaridade com o *embbedding* do usuário.

Algorithm 1 Geração de Explicação com Embeddings

```
1:  function CAMINHO_EMBEDDING(profile_items, rec_item, embed_model)
2:      user_embed ← sum(embed_model(profile_items))
3:      caminhos ← dijkstra(profile_items, rec_item)
4:      max ← −1
5:      cam_max ← []
6:      for c in caminhos do
7:          caminho_embed ← sum(embed_model(c.nos()))
8:          sim ← cosseno(caminho_embed, user_embed)
9:          if sim > max then
10:             max ← sim
11:             cam_max ← c
12:         end if
13:     end for
14:     return cam_max
15: end function
```

A proposta de *embedding* foi feita gerando *embeddings* dos GCs extraídos da Wikidata LOD para os domínios de filmes e artísticos das bases de dados MovieLens e LastFM, respectivamente. A escolha dos algoritmos de *embeddings* utilizados foi realizada considerando as diferentes famílias de algoritmos. Enquanto o TransE [18] e RotatE [29] utilizam a abordagem translacional, o algoritmo ComplEX [32], por sua vez, é um algoritmo bilinear. O algoritmo TransE foi comparado ao RotatE para verificar se a evolução do estado da arte na representação de *embeddings* de grafos em uma família de algoritmos, melhora métricas de qualidade de explicação.

Modelos translacionais se baseiam no conceito de coordenadas cartesianas [5, 36] em que, considerando uma tripla (h, r, t), onde h e t são nós no GC e r é a relação que conecta os dois nós, uma transformação linear que representa a distância entre esses elementos gera a função de custo a ser minimizada, assim, $h + r \approx t$. O RotatE usa a equação de distância $d_r(h, t) = ||h \circ r - t||$, já o TransE utiliza a função $d_r(h, t) = ||h + r - t||$. O símbolo $\circ$ denota o produto elemento a elemento. Modelos bilineares, por sua vez, medem a similaridade de h, r e t utilizando produtos interno e operações multiplicativas, geralmente envolvendo formas bilineares [17].

Para o treinamento dos modelos de *embedding* de grafos foi realizada a otimização de parâmetros em que a taxa de aprendizado (λ)

foi variada entre os valores 0.1 e 0.001, e o tamanho do *batch* (B) para atualização dos parâmetros foi de 128 e 256. O tamanho do *embedding* (K) também foi otimizado entre 200 e 400. A utilização ou não de amostragem negativa também foi variada. Quando existia a amostragem negativa, 10 exemplos negativos por amostra positiva foram gerados para um *batch*. A quantidade de épocas para o treinamento foi fixa de 40 para todos os modelos. Como o modelo ComplEX [32] usa o algoritmo de *Stochastic Gradient Descent* com AdaGrad [11] como otimizador, a taxa de aprendizado é ajustada no treinamento.

As triplas do GC foram divididas em conjuntos de treinamento, validação e teste na proporção de 0.8, 0.1 e 0.1, respectivamente. No GC extraído da Wikidata para os itens do conjunto de dados MovieLens, 235,466 triplas foram utilizadas para treino, 29,434 para validação e 29,433 para teste. Já para o GC extraído para o LastFM, a separação treino, validação e teste foi de 101,516, 12,690 e 12,689, respectivamente. A biblioteca Pykeen [1] foi utilizada para implementar os modelos de *embeddings* de grafo. A acurácia do modelo é medido pela métrica de *Hit Rate* que mede o quão bem o modelo encontra o nó que completa uma tripla. Assim, dado um *embedding* de nó h e um *embedding* de relação r, o modelo deve encontrar o *embedding* de nó t correto correspondente a tripla (h, r, t) presente no grafo. As métricas de *Hit Rate* no conjunto de teste para os modelos de melhores parâmetros para os *embeddings* de GC extraído da Wikidata para o MovieLens e LastFM estão apresentadas na Tabela 2.

Os melhores modelos obtiveram os seguintes parâmetros: Para o conjunto de dados MovieLens 100k, no modelo TransE K foi de 200, λ de 0.001, B de 256 e sem a presença de amostragem negativa; para o modelo ComplEX K foi de 400, B de 128 e com a presença de amostragem negativa; no modelo RotatE K foi de 200, λ de 0.001, B de 128 e com a presença de amostragem negativa. Para o conjunto de dados LastFM, no modelo TransE K foi de 200, λ de 0.001, B de 256 e sem a presença de amostragem negativa; para o modelo ComplEX K foi de 200, B de 128 e sem a presença de amostragem negativa; no modelo RotatE K foi de 200, λ de 0.001, B de 128 e com a presença de amostragem negativa.

		TransE	RotatE	ComplEX
MovieLens	H@1	0.0317	0.0982	0.0115
	H@3	0.0956	0.1595	0.0209
	H@5	0.1282	0.1927	0.0261
	H@10	0.1727	0.2418	0.0362
LastFM	H@1	0.0570	0.1852	0.0029
	H@3	0.1135	0.2725	0.0076
	H@5	0.1481	0.3154	0.0118
	H@10	0.1998	0.3735	0.0219

Tabela 2: Métricas do conjunto de teste para diferentes algoritmos de *embeddings* de grafos para o GC dos conjuntos de dados MovieLens e LastFM. $H@n$ é a métrica de Hit Rate do modelo para completar corretamente uma tripla considerando os n nós mais próximos.

3.3 Abordagens Sintáticas

Três algoritmos sintáticos foram implementados para comparação com os quatro resultados obtidos por meio da aplicação do método de *embedding* de grafos no arcabouço da descrito na Seção 3.2. Diferentemente de métodos baseados em *embeddings*, abordagens sintáticas utilizam a ocorrência do nó do atributo relacionado ao nó do item para determinar sua relevância.

O método ExpLOD [19] classifica propriedades no Grafo de Conhecimento (GC) usando uma abordagem adaptada de TF-IDF. A relevância de um atributo é determinada pela frequência de referências ao atributo tanto de itens interagidos quanto de itens recomendados em relação ao total de referências ao atributo entre todos os itens. A Equação 6 ilustra o cálculo para o valor de relevância de um atributo p, onde n_{p,I_u} representa o número de links do conjunto de itens interagidos pelo usuário I_u para o atributo p. Da mesma forma, n_{p,I_r} denota o número de links conectando o atributo p aos itens recomendados I_r, e $IDF(p)$ significa a Frequência Inversa do Documento de p, calculada como $\log(\frac{|C|}{n_{p,I_C}})$, onde $|C|$ representa o número total de itens no catálogo e n_{p,I_C} é o número total de itens referenciando o atributo p. Os valores α e β são pesos e foram definidos como 0.5 de acordo com [19].

$$score\_explod(p, I_u, I_r) = (\alpha \frac{n_{p,I_u}}{|I_u|}) + (\beta \frac{n_{p,I_r}}{|I_r|}) * IDF(p) \quad (6)$$

A Equação 7 exibe o cálculo para classificação de atributos do ExpLOD v2 [20], que é muito similar à sua versão anterior; mas também abrange atributos mais amplos. Por exemplo, considere o filme 'La La Land' de 2016, classificado com o atributo 'romance' no GC do Wikidata. Essa classificação implica que o filme também está associado a atributos mais amplos como 'relacionamento interpessoal' e 'amor', já que 'romance' é uma subclasse desses atributos. Como resultado, para atributos do GC mais amplos b que têm subclasses, a relevância é a soma da Equação 6 para todos os atributos p_{bi} no conjunto de $P_c(b)$ que são filhos de b, multiplicado pelo IDF da classe mais ampla ($IDF(b)$). Portanto, os algoritmos ExpLOD classificam atributos que são populares entre o conjunto de itens interagidos, mas raros no conjunto de itens do catálogo.

$$score\_explod(b, I_u, I_r) = \sum_{i=1}^{|P_c(b)|} score\_explod(p_{bi}, I_u, I_r) * IDF(b) \quad (7)$$

O mecanismo de pontuação usado no Modelo de Explicação Baseado em Propriedades (PEM) é representado pela Equação 8 e, diferentemente dos algoritmos ExpLOD, considera o número de itens interagidos que referenciam o atributo em vez do número de links. Para pontuar um atributo p, primeiramente, considera-se o número de itens interagidos que referenciam o atributo $|\overline{I(p, I_u)}|$, onde I_u representa o conjunto de itens com os quais o usuário interagiu. Esse valor é então normalizado pelo número total de itens com os quais o usuário interagiu, denotado por $|I_u|$.

Além disso, a equação considera o número de itens no catálogo C conectados ao atributo $|\overline{I(p, C)}|$. Semelhante ao termo anterior, esse valor é normalizado pelo número total de itens no catálogo, denotado por $|C|$. Finalmente, o logaritmo do número total de itens no

catálogo conectados ao atributo $\log(|I(p,C)|)$ é calculado para amplificar a importância de atributos relativamente raros no catálogo.

$$score\_pem(p, I_u, I_r, C) = \frac{|\overline{I(p, I_u)}|/|I_u|}{|\overline{I(p, C)}|/|C|} * \log(|I(p, C)|) \quad (8)$$

Para os todos algoritmos o caminho com a maior média de relevância de atributo é escolhido como explicação. Além disso, o comprimento máximo do caminho foi definido como cinco, e o número máximo de itens interagidos para um item recomendado foi três.

3.4 Métricas

A métrica de Recência da Interação de Conexão (*LIR*) mede a recência dos itens interagidos pelo usuário que formam uma explicação; a Popularidade da Entidade Compartilhada (*SEP*) mede a popularidade dos atributos exibidos em explicações para um único usuário e Diversidade do Tipo de Explicação (*ETD*) o número de atributos diferentes nas explicações. Assim, as métricas propostas por [2] para avaliar a qualidade de explicação definem que explicações de qualidade encontram caminhos diferentes (*ETD*), mas utilizando atributos populares (*SEP*) e conectando itens recomendados com itens recém interagidos (*LIR*). Todas as métricas variam entre 0 e 1 em que 1 é o valor ótimo, exceto por ETD pode ser maior que 1 caso o caminho tenha mais que um atributo.

As Equações 9 e 10 representam as métricas de *LIR* e *SEP*, respectivamente. Essas métricas são calculadas com base na média de equações de média móvel exponencialmente ponderada normalizada para cada item interagido e atributo dentro de uma explicação.

Para *LIR*, os valores dos itens interagidos (p^i) são calculados usando seus respectivos carimbos de data/hora (t^i), que são normalizados pelo método min-max para variar entre 0 e 1. A natureza recursiva da função garante que o valor de uma propriedade i depende de $i-1$, com valores ordenados em ordem crescente de carimbos de data/hora. Assim, $LIR(p^1, t^1)$ equivale a t_1. O parâmetro β é tipicamente definido como 0,3, como sugerido em [2]. Consequentemente, *LIR* atribui valores mais altos a explicações que conectam recomendações com itens interagidos mais recentes.

$$LIR(p^i, t^i) = (1-\beta) * LIR(p^{i-1}, t^{i-1}) + \beta * t^i \quad (9)$$

Na Equação 10, a métrica de Popularidade da Entidade Compartilhada (*SEP*) quantifica a popularidade dos atributos considerando o número v^i de nós de itens conectados ao atributo e^i. A ordenação e a normalização min-max também são aplicadas ao número de referências que um atributo têm a outros nós. Consequentemente, $SEP(e^1, v^1)$ corresponde a v_1, representando o atributo com o menor número de referências no GC. Valores altos de *SEP* indicam que os atributos em explicações são populares.

$$SEP(e^i, v^i) = (1-\beta) * SEP(e^{i-1}, v^{i-1}) + \beta * v^i \quad (10)$$

Finalmente, a Equação 11 define a métrica de Diversidade do Tipo de Explicação (*ETD*), que quantifica a diversidade de atributos em explicações associadas a recomendações. Ela calcula a razão do número de propriedades na lista recomendada $|\omega_{L_u}|$ para o mínimo entre o comprimento da lista de recomendação k e o número total de possíveis atributos ω_L que poderiam formar uma explicação. *ETD* fornece uma visão sobre a variedade de atributos apresentados em explicações e ajuda a avaliar se o algoritmo de explicação tende a favorecer atributos repetitivos. Valores mais altos de *ETD* indicam uma maior diversidade de atributos.

$$ETD(S) = \frac{|\omega_{L_u}|}{min(k, |\omega_L|)} \quad (11)$$

4 RESULTADOS

Para responder às questões de pesquisa QP1 e QP2, executamos os algoritmos sintáticos ExpLOD, ExpLOD v2, PEM e a abordagem de *embedding* com os algoritmos TransE, RotatE e ComplEX para todos os usuários dos conjuntos de dados MovieLens 100k e LastFM para os cinco principais itens recomendados dos algoritmos de recomendação Mais Popular, BPR-MF, PageRank, UserKNN, EASE e NCF considerando as métricas LIR, ETD e SEP. Os resultados para outras métricas, tais como precisão e diversidade, e exemplos de explicações gerados por cada um dos algoritmos estão disponíveis online[2].

A Tabela 3 e Tabela 4 correspondem à média e o desvio padrão das métricas de qualidade de explicação SEP, ETD e LIR para todos os usuários considerando os conjuntos de dados MovieLens 100k e LastFM, respectivamente, em relação aos cinco principais itens recomendados de cada algoritmo. As duas primeiras colunas são relativas ao algoritmo de recomendação executado e a métrica de qualidade, em seguida, os resultados do método proposto considerando três algoritmos de *embedding* diferentes estão nas três primeiras colunas e os resultados dos três métodos sintáticos estão nas últimas três colunas. Os valores em negrito são os mais altos entre os algoritmos, e os valores sublinhados são os mais baixos. Distinguimos os valores mais altos e mais baixos, porque os objetivos de explicação e os atributos de qualidade exibem um *trade-off*. Os atributos dos itens são distribuídos seguindo uma *long-tail* em que alguns atributos são muito comuns entre poucos itens [13] e, assim, quanto maior a quantidade de atributos mostrados em explicações aos usuários, maior também é a probabilidade de um atributo menos popular de ser escolhido em uma explicação [2, 3, 31].

A fim de responder a QP1 e diferenciar o impacto de diferentes algoritmos de *embeddings* na geração de explicações, as três primeiras colunas das tabelas se referem os resultados das qualidade de explicações do método agnóstico ao modelo utilizando os algoritmos de *embedding* de grafos TransE [18], RotatE [29] e ComplEX [32].

Considerando os métodos TransE [18] e RotatE [29], que pertencem a família de algoritmos translacionais de *embedding* de grafos, as métricas de *HitRate* no conjunto de treinamento do modelo de *embedding* de grafos na Tabela 2 evidenciam que o algoritmo RotatE [29] é uma evolução no estado da arte para o modelo translacional TransE [18]. Nesse sentido, no conjunto de dados LastFM esta evolução do estado-da-arte refletiu na evolução das métricas de qualidade de explicação, em que tanto ETD quanto SEP obtiveram resultados melhores para o método com o algoritmo RotatE em comparação ao TransE. No entanto, o mesmo não ocorreu no dataset do MovieLens, em que a melhoria das métricas de *HitRate* de um mesmo tipo de modelo não necessariamente refletiu na evolução de métricas de qualidade de explicação. Assim, mesmo modelos mais simples de *embedding* de grafos conseguem gerar representações vetoriais que refletem em explicações de qualidade.

[2] https://tinyurl.com/zfscfshv

		TransE	RotatE	ComplEX	ExpLOD	ExpLOD v2	PEM
Mais Popular	LIR	0.03147 ± 0.12	0.0275 ± 0.11	0.0234 ± 0.09	**0.0945 ± 0.15**	0.0834 ± 0.14	0.0320 ± 0.07
	ETD	0.6718 ± 0.21	0.6842 ± 0.20	**0.9537 ± 0.31**	0.5809 ± 0.19	0.5947 ± 0.19	0.9390 ± 0.12
	SEP	0.52104 ± 0.18	**0.6869 ± 0.13**	0.5625 ± 0.15	0.6322 ± 0.14	0.6107 ± 0.12	0.1430 ± 0.13
Page Rank	LIR	0.0310 ± 0.11	0.0312 ± 0.12	0.0241 ± 0.10	**0.0939 ± 0.15**	0.0872 ± 0.15	0.0323 ± 0.08
	ETD	0.7335 ± 0.21	0.6570 ± 0.22	0.9305 ± 0.31	0.5563 ± 0.20	0.6043 ± 0.19	**0.9389 ± 0.12**
	SEP	0.4662 ± 0.20	**0.7022 ± 0.15**	0.5557 ± 0.16	0.6250 ± 0.16	0.5606 ± 0.15	0.1095 ± 0.11
UserKNN	LIR	0.03327 ± 0.12	0.0352 ± 0.12	0.0234 ± 0.10	**0.1010 ± 0.14**	0.0914 ± 0.13	0.0322 ± 0.08
	ETD	0.7055 ± 0.24	0.6849 ± 0.22	**0.9859 ± 0.33**	0.6593 ± 0.19	0.6409 ± 0.19	0.9452 ± 0.11
	SEP	0.5202 ± 0.23	0.2311 ± 0.15	**0.6103 ± 0.16**	0.5803 ± 0.16	0.5275 ± 0.14	0.131 ± 0.12
BPR-MF	LIR	0.0408 ± 0.16	0.0298 ± 0.11	0.0244 ± 0.09	**0.1048 ± 0.14**	0.0945 ± 0.13	0.0306 ± 0.07
	ETD	0.6826 ± 0.26	0.6934 ± 0.24	**0.9855 ± 0.32**	0.6891 ± 0.19	0.6937 ± 0.19	0.9583 ± 0.10
	SEP	0.5755 ± 0.23	0.2151 ± 0.15	0.5013 ± 0.16	**0.6113 ± 0.14**	0.5428 ± 0.14	0.1400 ± 0.12
EASE	LIR	0.0312 ± 0.12	0.0295 ± 0.11	0.0209 ± 0.09	**0.1000 ± 0.14**	0.0912 ± 0.14	0.03193 ± 0.08
	ETD	0.7345 ± 0.24	0.6751 ± 0.23	**0.9724 ± 0.31**	0.6204 ± 0.20	0.6328 ± 0.19	0.9459 ± 0.11
	SEP	0.4952 ± 0.49	**0.6583 ± 0.17**	0.6089 ± 0.16	0.5743 ± 0.17	0.5274 ± 0.15	0.1350 ± 0.13
NCF	LIR	0.0320 ± 0.13	0.0244 ± 0.09	0.0199 ± 0.08	**0.1181 ± 0.13**	0.1035 ± 0.12	0.0380 ± 0.08
	ETD	0.6966 ± 0.29	0.8080 ± 0.25	**1.0100 ± 0.32**	0.8432 ± 0.16	0.8161 ± 0.16	0.9885 ± 0.05
	SEP	**0.6122 ± 0.22**	0.2511 ± 0.14	0.3955 ± 0.14	0.5868 ± 0.14	0.5350 ± 0.13	0.1613 ± 0.11

Tabela 3: Tabela das métricas LIR, ETD e SEP para explicações dos cinco principais itens recomendados no conjunto de dados MovieLens 100k. Valores em negrito são os mais altos e valores sublinhados são os mais baixos entre os algoritmos de recomendação.

		TransE	RotatE	ComplEX	ExpLOD	ExpLOD v2	PEM
Mais Popular	LIR	0.0104 ± 0.06	0.0100 ± 0.06	0.0123 ± 0.08	0.0182 ± 0.09	**0.0189 ± 0.11**	0.0143 ± 0.08
	ETD	0.8247 ± 0.27	0.9319 ± 0.25	**1.1394 ± 0.28**	0.7023 ± 0.17	0.4927 ± 0.22	0.9212 ± 0.12
	SEP	0.5084 ± 0.22	0.6617 ± 0.21	0.5181 ± 0.16	0.7097 ± 0.17	**0.7537 ± 0.23**	0.1214 ± 0.08
Page Rank	LIR	0.0108 ± 0.07	0.008 ± 0.06	0.0125 ± 0.07	0.0191 ± 0.10	**0.0212 ± 0.12**	0.0134 ± 0.07
	ETD	0.7683 ± 0.25	0.8704 ± 0.30	**1.0769 ± 0.32**	0.6100 ± 0.20	0.5447 ± 0.19	0.9440 ± 0.10
	SEP	0.4820 ± 0.18	0.6359 ± 0.17	0.5022 ± 0.18	0.6501 ± 0.21	**0.7164 ± 0.21**	0.1209 ± 0.09
UserKNN	LIR	0.0102 ± 0.07	0.0090 ± 0.05	0.0117 ± 0.07	0.0183 ± 0.10	**0.0191 ± 0.11**	0.0158 ± 0.08
	ETD	0.7760 ± 0.26	0.8688 ± 0.31	**1.0624 ± 0.32**	0.5335 ± 0.20	0.5355 ± 0.18	0.9106 ± 0.14
	SEP	0.5071 ± 0.19	**0.6088 ± 0.18**	0.4540 ± 0.18	0.5288 ± 0.26	0.2810 ± 0.22	0.1417 ± 0.10
BPR-MF	LIR	0.0110 ± 0.06	0.0096 ± 0.06	0.0113 ± 0.07	0.0191 ± 0.10	**0.0204 ± 0.11**	0.0164 ± 0.08
	ETD	0.8403 ± 0.26	0.9219 ± 0.31	**1.1021 ± 0.31**	0.6145 ± 0.21	0.6196 ± 0.19	0.9450 ± 0.11
	SEP	0.5312 ± 0.18	0.6002 ± 0.17	0.5984 ± 0.18	0.5605 ± 0.23	**0.6302 ± 0.19**	0.1759 ± 0.12
EASE	LIR	0.0102 ± 0.07	0.0092 ± 0.05	0.0111 ± 0.07	0.0188 ± 0.11	**0.0194 ± 0.11**	0.0154 ± 0.08
	ETD	0.7934 ± 0.25	0.8753 ± 0.32	**1.0707 ± 0.32**	0.5474 ± 0.20	0.5585 ± 0.18	0.9246 ± 0.13
	SEP	0.5026 ± 0.19	**0.5870 ± 0.19**	0.4639 ± 0.18	0.5307 ± 0.25	0.2861 ± 0.21	0.1466 ± 0.10
NCF	LIR	0.0098 ± 0.06	0.0106 ± 0.06	0.0131 ± 0.07	**0.0182 ± 0.09**	0.0162 ± 0.08	0.0162 ± 0.08
	ETD	0.9409 ± 0.27	1.0318 ± 0.32	**1.2057 ± 0.29**	0.7775 ± 0.18	0.7867 ± 0.18	0.9589 ± 0.09
	SEP	0.6302 ± 0.17	**0.6509 ± 0.16**	0.6153 ± 0.17	0.5904 ± 0.19	0.5514 ± 0.20	0.2748 ± 0.15

Tabela 4: Tabela das métricas LIR, ETD e SEP para explicações dos cinco principais itens recomendados no conjunto de dados LastFM. Valores em negrito são os mais altos entre os algoritmos de recomendação e valores sublinhados são os mais baixos.

Nesse sentido, o algoritmo ComplEX [32], da família de algoritmos bilineares de *embedding* de grafos, obteve os resultados mais consistentes para ambas as bases. Particularmente este algoritmo obteve ETD acima de 0.65 e SEP acima de 0.45 para todas os conjuntos de dados e algoritmos. Isso acontece, pois esses modelos utilizam formas bilineares para gerar as representações vetoriais de grafos, e isso possibilita a modelagem de padrões mais complexos entre nós e arestas. Diferentemente, modelos translacionais geram *embeddings* por meio da aproximação dos vetores realizando translações, limitando sua expressividade. Dessa forma, respondendo a QP1, a utilização de um tipo de modelo de *embedding* que captura

relações mais complexas entre nós e arestas refletiu na melhora de métricas de qualidade de explicação.

Para responder a QP2 e comparar as diferenças entre abordagens sintáticas e semânticas, consideramos também os modelos ExpLOD [19], ExpLOD v2 [20] e PEM [10] (nas três últimas colunas da Tabela 3 e Tabela 4). Neste contexto, é possível perceber que, para ambos os conjuntos de dados, os algoritmos sintáticos, especialmente os algoritmos de ExpLOD e do ExpLOD v2, foram as melhores entre todos os outros algoritmos para as métricas LIR e SEP. Por outro lado, as abordagens de *embeddings* foram superiores na métrica ETD de diversidade.

No caso do algoritmo RotatE, por exemplo, para os algoritmos de Mais Popular e PageRank no conjunto de dados MovieLens e UserKNN, BPR-MF, EASE e NCF no LastFM, as métricas de SEP e ETD foram melhores quando comparadas à algoritmos sintáticos. Isso indica a capacidade de métodos de *embedding* de fornecer caminhos de explicação diversos para várias recomendações e manter um certo nível de popularidade de atributos. No entanto, algoritmos de *embeddings* também demonstraram menores níveis de LIR, o que pode ser atribuído à sua metodologia de treinamento. Embora identifique caminhos diversos para explicações de diferentes recomendações, o algoritmo incorpora o item interagido apenas no processo de soma de *pooling* para gerar os *embeddings* do caminho de explicação e do usuário. Em contraste, os três métodos sintáticos priorizam a quantidade de nós de itens interagidos que estão conectados a nós de atributos no GC para a escolha do caminho mais relevante a ser mostrado como explicação.

A exceção é o algoritmo PEM, que reflete o comportamento de *trade-off* entre diversidade e popularidade de atributos. Este algoritmo alcança uma diversidade muito alta, mas, em contraste, possui a menor SEP para todas as métricas. Isso acontece porque, ao contrário dos algoritmos ExpLOD, que são baseados em TF-IDF, o PEM normaliza o número de itens que referenciam um atributo aos itens interagidos pelo catálogo. Como o catálogo de itens é extenso, o PEM é mais suscetível a exibir mais itens diversos.

Portanto, algoritmos sintáticos definem a relevância do caminho de explicação como um *trade-off* entre o número de conexões dos nós de atributos com nós de itens. Nesses algoritmos a explicação escolhida contém nós de atributos que estão conectados a muitos nós de itens interagidos pelo usuário, mas pouco conectados a nós do conjunto inteiro de itens. Em particular, os algoritmos ExpLOD [19] e ExpLOD v2 [20] priorizam a popularidade, enquanto o PEM [10] prioriza a diversidade. Já os modelos de *embeddings* são treinados na tarefa de completar triplas dos grafos. Assim, considerando um nó h e uma relação r, algoritmos de *embedding* de grafos aprendem a encontrar o nó correto t da tripla (h, r, t) do GC. Consequentemente, algoritmos de explicação que empregam representações vetoriais são balanceados considerando as métricas de SEP e ETD.

Respondendo a QP2, observamos que métodos sintáticos estão mais propensos a serem influenciados pela popularidade, porque escolhem explicações considerando o número de ligações de um nó de atributo aos nós de itens. Por outro lado, os métodos baseados em *embeddings* escolhem caminhos de explicação a partir da similaridade da representação vetorial de nós e arestas do grafo. Logo, para estas estratégias, a popularidade de nós de atributos em nós itens não interfere decisivamente na escolha da explicação a um item recomendado, sendo, dessa maneira, mais equilibrados e, consequentemente, melhores que métodos sintáticos.

5 CONCLUSÕES

Neste trabalho foi desenvolvida uma abordagem comparativa e reprodutível, analisando o efeito de diferentes algoritmos de *embeddings* de grafos na geração de explicações de qualidade em algoritmos agnósticos a modelos em SsR, utilizando GC. Dessa maneira, por meio de três métricas de explicabilidade: a recência dos itens e diversidade e popularidade de atributos, as explicações de cada algoritmo foi avaliada para duas bases de dados e seis SsR.

Foi verificado que abordagens baseadas em *embeddings* conseguem balancear melhor a popularidade e diversidade de atributos comparando com abordagens sintáticas. Além disso, métricas de treinamento de diferentes métodos de *embedding* não necessariamente refletem na melhoria nas métricas de qualidade de explicação. Como trabalhos futuros, modelos geracionais podem ajudar na otimização multi-parâmetro e considerar todas as três métricas de qualidade, visto que algoritmos sintáticos de explicação priorizam popularidade ou diversidade de atributos e modelos de *embedding* não priorizam a recência dos itens.

AGRADECIMENTOS

Os autores agradecem à CAPES, CNPq, Fapesp, AWS e Fapemig pelo financiamento e apoio a esta pesquisa.

REFERÊNCIAS

[1] Ali, M., Berrendorf, M., Hoyt, C.T., Vermue, L., Sharifzadeh, S., Tresp, V., Lehmann, J.: PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. Journal of Machine Learning Research **22**(82), 1–6 (2021), http://jmlr.org/papers/v22/20-825.html

[2] Balloccu, G., Boratto, L., Fenu, G., Marras, M.: 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. pp. 646–656 (2022)

[3] Balog, K., Radlinski, F.: Measuring recommendation explanation quality: The conflicting goals of explanations. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. pp. 329–338 (2020)

[4] Cantador, I., Brusilovsky, P., Kuflik, T.: 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: Proceedings of the 5th ACM conference on Recommender systems. RecSys 2011, ACM, New York, NY, USA (2011)

[5] Cao, J., Fang, J., Meng, Z., Liang, S.: Knowledge graph embedding: A survey from the perspective of representation spaces. ACM Computing Surveys **56**(6), 1–42 (2024)

[6] Coba, L., Confalonieri, R., Zanker, M.: Recoxplainer: A library for development and offline evaluation of explainable recommender systems. IEEE Computational Intelligence Magazine **17**(1), 46–58 (2022). https://doi.org/10.1109/mci.2021.3129958

[7] Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems. p. 39–46. RecSys '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1864708.1864721, https://doi.org/10.1145/1864708.1864721

[8] Da Costa, A., Fressato, E., Neto, F., Manzato, M., Campello, R.: Case recommender: a flexible and extensible python framework for recommender systems. In: Proceedings of the 12th ACM Conference on Recommender Systems. pp. 494–495 (2018)

[9] Dijkstra, E.W., et al.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)

[10] Du, Y., Ranwez, S., Sutton-Charani, N., Ranwez, V.: Post-hoc recommendation explanations through an efficient exploitation of the dbpedia category hierarchy. Knowledge-Based Systems **245**, 108560 (2022)

[11] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research **12**(7) (2011)

[12] Ferrari Dacrema, M., Boglio, S., Cremonesi, P., Jannach, D.: A troubling analysis of reproducibility and progress in recommender systems research. ACM Transactions on Information Systems (TOIS) **39**(2), 1–49 (2021)
[13] Ferraro, A.: Music cold-start and long-tail recommendation: bias in deep representations. In: Proceedings of the 13th ACM conference on recommender systems. pp. 586–590 (2019)
[14] Hada, D.V., Shevade, S.K.: Rexplug: Explainable recommendation using plug-and-play language model. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 81–91 (2021)
[15] Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis) **5**(4), 1–19 (2015)
[16] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web. p. 173–182. WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2017). https://doi.org/10.1145/3038912.3052569
[17] Li, J., Yang, Y.: Star: Knowledge graph embedding by scaling, translation and rotation. In: International Conference on AI and Mobile Services. pp. 31–45. Springer (2022)
[18] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the AAAI conference on artificial intelligence. vol. 29 (2015)
[19] Musto, C., Narducci, F., Lops, P., De Gemmis, M., Semeraro, G.: Explod: a framework for explaining recommendations based on the linked open data cloud. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 151–154 (2016)
[20] Musto, C., Narducci, F., Lops, P., de Gemmis, M., Semeraro, G.: Linked open data-based explanations for transparent recommender systems. International Journal of Human-Computer Studies **121**, 93–107 (2019)
[21] Peng, C., Xia, F., Naseriparsa, M., Osborne, F.: Knowledge graphs: Opportunities and challenges. Artificial Intelligence Review pp. 1–32 (2023)
[22] Rana, A., D'Addio, R.M., Manzato, M.G., Bridge, D.: Extended recommendation-by-explanation. User Modeling and User-Adapted Interaction **32**(1-2), 91–131 (2022)
[23] Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. p. 452–461. UAI '09, AUAI Press, Arlington, Virginia, USA (2009)
[24] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work. p. 175–186. CSCW '94, Association for Computing Machinery, New York, NY, USA (1994). https://doi.org/10.1145/192844.192905, https://doi.org/10.1145/192844.192905
[25] Ricci, F., Rokach, L., Shapira, B.: Recommender systems: introduction and challenges. Recommender systems handbook pp. 1–34 (2015)
[26] Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence **1**(5), 206–215 (2019)
[27] Souza, L.S.d., Manzato, M.G.: Aspect-based summarization: an approach with different levels of details to explain recommendations. In: Proceedings of the Brazilian Symposium on Multimedia and the Web. pp. 202–210 (2022)
[28] Steck, H.: Embarrassingly shallow autoencoders for sparse data. In: The World Wide Web Conference. p. 3251–3257. WWW '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3308558.3313710, https://doi.org/10.1145/3308558.3313710
[29] Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv preprint arXiv:1902.10197 (2019)
[30] Tchuente, D., Lonlac, J., Kamsu-Foguem, B.: A methodological and theoretical framework for implementing explainable artificial intelligence (xai) in business applications. Computers in Industry **155**, 104044 (2024)
[31] Tintarev, N., Masthoff, J.: Explaining recommendations: Design and evaluation. In: Recommender systems handbook, pp. 353–382. Springer (2015)
[32] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
[33] Xu, Z., Zeng, H., Tan, J., Fu, Z., Zhang, Y., Ai, Q.: A reusable model-agnostic framework for faithfully explainable recommendation and system scrutability. ACM Transactions on Information Systems (2023)
[34] Zanon, A.L., da Rocha, L.C.D., Manzato, M.G.: Balancing the trade-off between accuracy and diversity in recommender systems with personalized explanations based on linked open data. Knowledge-Based Systems **252**, 109333 (2022)
[35] Zanon, A.L., da Rocha, L.C.D., Manzato, M.G.: Model-agnostic knowledge graph embedding explanations for recommender systems. In: World Conference on Explainable Artificial Intelligence. pp. 3–27. Springer (2024)
[36] Zhang, S., Tay, Y., Yao, L., Liu, Q.: Quaternion knowledge graph embeddings. Advances in neural information processing systems **32** (2019)
[37] Zhang, Y., Chen, X., et al.: Explainable recommendation: A survey and new perspectives. Foundations and Trends® in Information Retrieval **14**(1), 1–101 (2020)