A Fairness Analysis of GPT-2's Interpretability with GNN-Generated Knowledge Graph Embeddings Delivered through Soft Prompts

Ahmad Khalidi[1], Thomas Clemen[1]

[1]Hamburg University of Applied Sciences (HAW Hamburg), 5 Berliner Tor, Hamburg, 20099

Abstract

Naive fairness criteria applied to machine learning (ML) models, such as low feature importance of sensitive features, are not sufficient criteria for formal group fairness. On the contrary, the inclusion of group membership can be justified in order to achieve group fairness. A currently relevant model architecture combining large language model (LLMs) and graph neural networks (GNNs) is suitable for investigating this phenomenon. Not only is this architecture showing promising results in the processing of knowledge graphs (KGs), it also combines two different algorithmic perspectives on underlying data.

In this paper, we stage a scenario in which a given training dataset contains systematic bias. We assume that when ML models are exposed to this bias during training, their fairness is affected. We propose that a fairness criterion with regard to this bias should take into account systematic processing within the models. We argue that if an ML model is exposed to systematic bias, it should be better able to process this systematic bias in a structured manner.

In a simple scenario using explainable AI (XAI) methods, we show that the hybrid GNN-LLM architecture processes systematic bias in a structured manner.

Keywords

LLM, Knowledge Graph, Graph Representation Learning, Explainable AI, Fairness, Soft Prompt

1. Introduction

LLMs such as Llama3 [1], GPT4 [2], Gemma2 [3] and DeepSeek [4] are already showing amazing results in natural language processing (NLP). The processing of structured texts, such as in KGs, is also being investigated further. In particular, the combination of GNNs and transformer models is enjoying great popularity and promising success [5, 6, 7]. However, this technology requires transparency and explainability in order to create trust and fairness and to comply with regulations [8].

XAI methods can be used under strict conditions for the investigation of fairness in LLMs. The basis for the successful application of XAI to measure fairness are suitable fairness criteria combined with a technical understanding of XAI methods and their significance [9]. The fairness criteria applied in this paper refer to group fairness. First, we demonstrate that a significant proportion of all input edges are directed toward a small number of nodes (its group membership). We then use XAI methods to show that this bias is interpreted in a structured manner in the GraphPrompter model. We argue that this property should be taken into account when evaluating the fairness of this model.

For our GraphPrompter implementation, we decided to use GraphSage [10] for the GNN and GPT2-Medium[1] [11] for the Transformer. As a down-stream task, we train the models on link prediction in the MovieLens dataset [12]. The structured form of the dataset allows us to design the applied XAI methods in a concept-based way [8, 13]. Our implementation also has tendencies of explainable-by-design, as the concept-based approach is trained in fine-tuning with the help of segments.

ECAI 2025: 1st International Workshop on eXplainable AI, Knowledge Representation and Knowledge Graphs, June 24–10, 2025, Bologna

ahmad.khalidi@haw-Hamburg.de (A. Khalidi); thomas.clemen@haw-hamburg.de (T. Clemen)

0009-0000-6640-7944 (A. Khalidi); 0000-0002-8200-5141 (T. Clemen)

[1]https://huggingface.co/openai-community/gpt2-medium last accessed: 08.06.25

2. Related Work

2.1. LLMs (Transformer)

In 2017, [14] proposed the application of the Transformer architecture in NLP and showed great success compared to the trends of sequential models. The architecture is largely based on the so-called multi-head-attention mechanism. This allows the learning of relationships between all positions within the texts and thus creates a semantic understanding of words and their context. The original form of this transformer architecture consisted of an encoder and decoder. The encoder was intended for embedding the input text in the vector space and the decoder for the autoregressive generation of the output text. Modern generative transformer architectures are usually based only on the decoder [1, 2, 3, 4]. **GPT2** [11] is a decoder-only model that can effectively be understood as a multitask model. The Hugging Face Transformers [15] implementation of GPT2[2] supports *text and positional encodings* as well as *segment encodings* as proposed in BERT [16].

2.2. Graph Representation Learning

Mapping graph structures to low-dimensional vectors [17, 18] allows back-end models to process the complex relationships between entities [19]. Since KGs encode structured factual knowledge with multirelational edges [20], multimodal nodes [21] and follow logical rules, constraints and ontologies, we call low-dimensional vector representations of KGs **Knowledge Graph Embeddings (KGEs)** [20]. KGs are structured data where entities are related to each other and facts (data points) are referred to as triple *(h,r,t)*, where h represents the head (source node), r the relationship (edge) between nodes and t the tail (target node) [22].
In the link prediction task, missing entities *(h,r,?)* or *(?,r,t)* are defined as fact triples [23]. Link prediction can be used for recommender systems [24, 25] or KG completion [26], among other tasks. This decision is often based on the similarity between the KGEs of two nodes, whereby nodes that share an edge are closer to each other than if they do not share an edge [27]. A typical measure for the similarity of two KGEs is the dot product [20], whereby cosine similarity is widely used [28].
GNNs often aggregate the node embeddings of their neighborhood using the message passing concept [18, 29]. **GraphSage** [10] interprets the message passing concept in the following way

$$h_i^{'} = W_1 h_i + W_2 * mean_{j \in N(i)} h_j$$

Where $h_i^{'}$ are the KGEs after aggregation over the sampled neighborhood $N(i)$ with the trainable weight matrices W_1 and W_2.
A combination of GNN and Transformer can happen under various aspects. The GraphPrompter architecture [7] interprets the GNN as an encoder and the Transformer as a predictor. The KGEs generated in the GNN are transferred to the Transformer via soft prompts [30] within the input embeddings. The trainable weights of the transformer are frozen during fine-tuning and only the weights of the GNN are trained. The authors of the paper demonstrate an increase in performance of this architecture in graph representation learning tasks, but do not show the internal learned strategies of the models.

In a Pytorch Geometrics (PyG) tutorial for Link Prediction[3], the authors implement a GraphSage model [10], and train it on the MovieLens dataset [12]. The model was trained to generate KGEs of nodes connected by an edge with high similarity and KGEs of nodes not connected by an edge with low similarity. During preprocessing of the MovieLens dataset, all users and movies are assigned consecutively numbered IDs. The tags are also ignored and the ratings are replaced with a label *has rated* and *has not rated*. The dataset is then split into *training, test* and *validation*, whereby some of the training data is excluded for message passing.

[2] https://huggingface.co/docs/transformers/v4.52.3/en/model_doc/gpt2#transformers.models.gpt2.modeling_gpt2
[3] Link accessed 09.08.25, released under MIT License: https://colab.research.google.com/drive/1xpzn1Nvai1ygd_P5Yambc_oe4VBPK_ZT?usp=sharing

2.3. XAI in Transformers

Despite their supposed neutrality, KGs exhibit biases [31]. These can manifest themselves in the models trained on them [32]. To strengthen trust in this technology, ensure fairness and comply with regulations, experts and laypersons must be able to understand and explain the underlying mechanisms and decision-making processes of AI models [8].
XAI methods can be divided into local and global methods, among others, in which the behavior of a model is described for a specific data point or in general [33]. With concept-based XAI methods, input texts can be mapped to concepts and allow us to summarize positions within the models [8, 13]. With classifier-based probing, a classifier can be trained on a specific task in order to reveal properties in the internal vector representation [8, 34].
T-SNE [35] maps high-dimensional vector representations to two- or three-dimensional vector representations. The mapping is non-linear and keeps vectors close to each other that are close to each other in high-dimensional space.
In [13] the authors combined local XAI methods, such as attention maps, with concept-based XAI methods to aggregate groups of concepts instead of individual items. These concept-based attention maps allow global insights to be generated in a further aggregation over the entire data set.

2.4. Fairness

A common fairness criteria is the absence of discrimination, which is often referred to as *statistical parity* [36, 37, 38]. In this context, individuals of a population are divided into groups based on sensitive features, like gender or race [9, 36, 39]. Statistical parity in ML is defined as the independence of *group membership* in predictions [36]. A common misconception is that removing sensitive features from the model improves their fairness. This misconception is refereed to as *fairness through unawareness*. Sometimes, group membership can be predicted from other attributes [9, 36, 37] or including group membership in the decision process has some legitimate and causal cause [9, 36].
Fair recommender systems operate in a field of tension between utility and exposure [40, 41, 42]. From the perspective of users, a fair recommender system provides them with a wide range of mainstream, niche and user specific information. From the perspective of providers, a fair recommender system exposes items equally, independent of the producers' popularity [41, 42]. Studies that examine the fairness of recommender systems often focus on the exposure of items with low popularity [41, 43, 44], mitigating the Matthew effect [45] and popularity bias [46].

3. Model Architecture

The GraphPrompter architecture [7] is based on a GNN and a Transformer model. The GNN generates KGEs based on a KG and transfers them to the Transformer. This configures the transferred KGEs into the input embeddings and generates an output based on this. Our implementation of the GNN in link prediction is based on the PyG tutorial for link prediction. For each link prediction, this means that two KGEs are produced by GNN and passed to the transformer, one for the start node and one for the target node.
We use the Hugging Face Transformers [15] implementation of GPT2-medium *SequenceClassification*[4] as the transformer. To do this, we expand the set of expected segments to include the concepts that we want to distinguish in our XAI methods. In fine-tuning, we train the transformer to use the segment embeddings, which, among other things, mark the positions of the KGEs. We base this idea on [13], in which the authors have a structured data set in which recurring elements are expected in each data point. Figure 1 describes the GraphPrompter architecture as we have implemented it for the MovieLens data set. The movies are reduced to the properties (*Movie-ID*, *Title* and *Genres*). Each position in the internal vector representation is assigned a segment according to its affiliation. For example, each position that encodes a separator token is assigned an arbitrary number *sep_id*. The fact quadruples

[4]https://huggingface.co/docs/transformers/v4.52.3/en/model_doc/gpt2#transformers.GPT2ForSequenceClassification

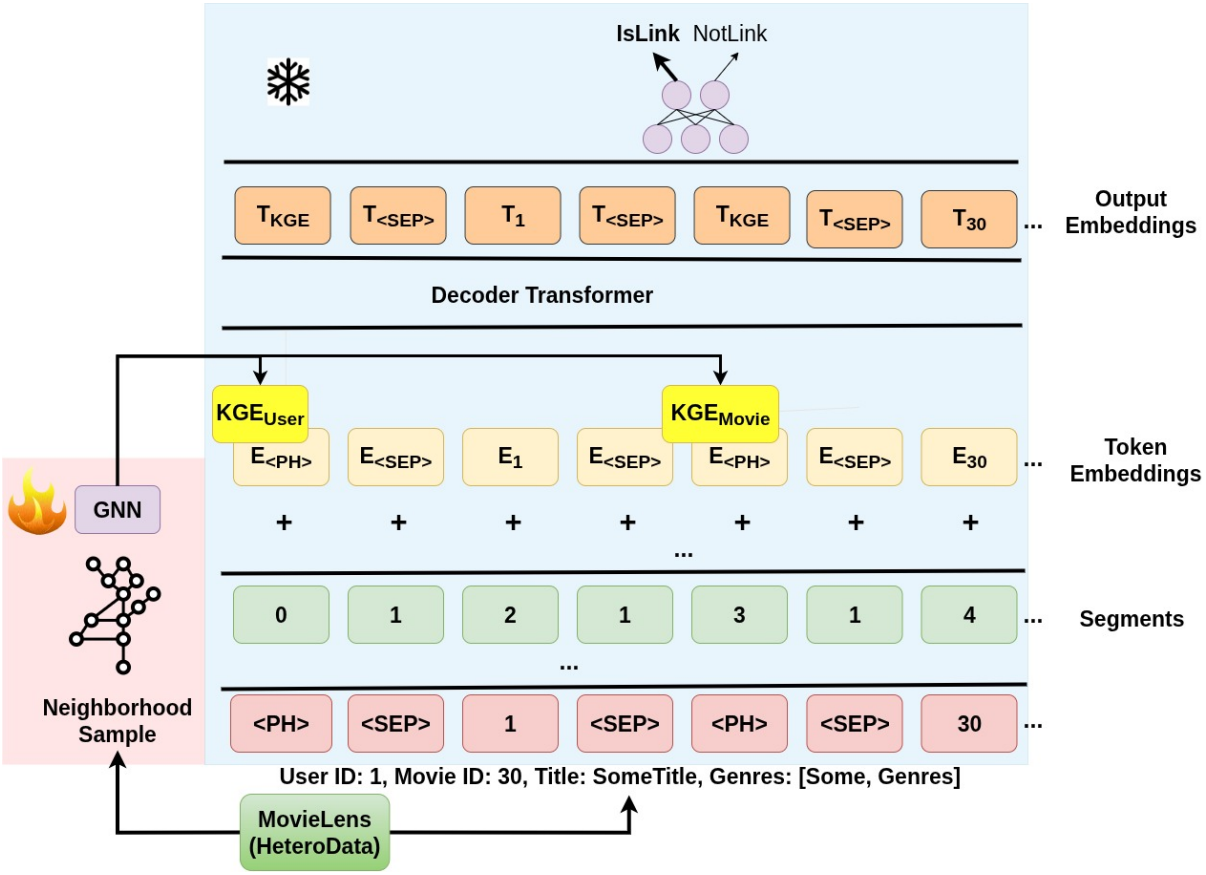


Figure 1: GraphPrompter architecture on the task of link prediction adapted from *Can we Soft Prompt LLMs for Graph Learning Tasks?*, by Lio et al., 2024 and *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, by Devlin et al., 2019

are reduced to the triples (*User ID*, *Movie ID*,*has[not]rated*). A request to GraphPrompter samples a neighborhood for a given data point. Based on this, the GraphSage model generates two KGEs, one for the user and one for the movie. The input positions are encoded in the Transformer. Instead of concatenation, we use placeholders, which we replace with the KGEs generated by GraphSage. The segments perform two tasks. 1) They act as a mask to replace the KGEs in a differentiable way with the placeholders. 2) They offset the embedding and thus offer the transformer a semantic differentiation of individual items within the input prompt.
Let $\vec{\mathbf{S}} \in \mathbb{R}^{d_B \times d_S}$ be the segments in batch and $\vec{\mathbf{W}} \in \mathbb{R}^{d_B \times d_S \times d_H}$ be word embeddings, while $d_B \in \mathbb{R}_{>0}$ is the batch length, $d_S \in \mathbb{R}_{>0}$ is the sequence length and $d_H \in \mathbb{R}_{>0}$ the hidden size ($d_H = 1024$ for *GPT2*), then we calculate the word-embedding $\vec{\mathbf{W}}$ that embeds the KGEs with

$$\vec{\mathbf{T}}_i = \begin{cases} 1, & \text{if } \vec{\mathbf{S}} = \mathbf{S}_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\vec{\mathbf{M}}_i = \vec{\mathbf{T}}_i \vec{\mathbf{1}}_{\mathbf{d}_H} \tag{2}$$

$$\overrightarrow{\mathbf{KGE}}_i = \overrightarrow{\mathbf{kge}}_i \vec{\mathbf{1}}_{d_S} \tag{3}$$

$$\vec{\mathbf{W}} = \left(\vec{\mathbf{W}} \odot (\vec{\mathbf{1}} - \vec{\mathbf{M}}_u) + \overrightarrow{\mathbf{KGE}}_u \odot \vec{\mathbf{M}}_u\right) \odot (\vec{\mathbf{1}} - \vec{\mathbf{M}}_m) + \overrightarrow{\mathbf{KGE}}_m \odot \vec{\mathbf{M}}_m \tag{4}$$

while
$i \in [u, m]$ is an index for either user or movie,
$\vec{\mathbf{S}}_i \in \mathbb{R}$ are the segments for user and movie (here 2 and 3),
$\vec{\mathbf{1}}_{d_H}$ is the unit vector of dimension d_H,

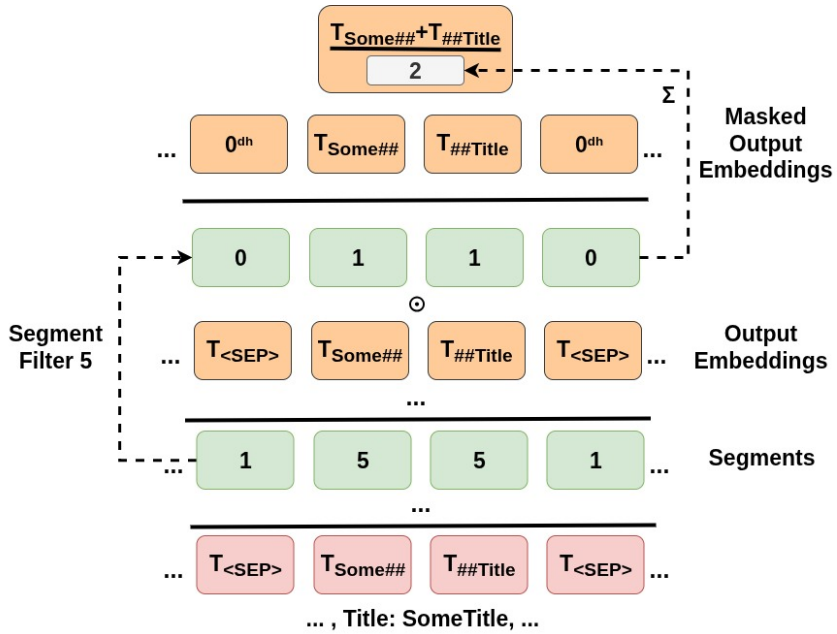


Figure 2: Segment-based aggregation of hidden states as an XAI method adapted from *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, by Devlin et al., 2019

$\vec{\mathbf{1}}_{d_S}$ is the unit vector of dimension d_S,
$\overrightarrow{\mathbf{kge}_i} \in \mathbb{R}^{d_B \times d_H}$ are the user and movie KGEs of dimension d_H distributed over the entire batch d_B and $\vec{\mathbf{1}}$ is the unit vector of dimension $(d_B \times d_S \times d_H)$.
In equation 1, we filter the segments in the batch by the segments of the KGEs and thus create a mask for the respective positions. Then we scale both segment masks $\vec{\mathbf{T}}$ via the hidden states d_H and both KGEs via the sequence length d_S in the equations 2 and 3 by repeating. In equation 4, we then superimpose the masked and scaled KGEs with the inverse-masked word embeddings and thus obtain the word embedding with the KGEs at the placeholder positions.

4. Segment-based XAI

In this paper, we first focus on the XAI methods *dimension-reduction* of the hidden states. As suggested in [13], concept-based XAI methods can be used to aggregate all positions of the same segments in order to obtain a semantic-oriented view. For the hidden states, we calculate the average of all positions in which the sequence is the same. Let SEG be the set of all segments, $i \in SEG$ an element from SEG, $d_L \in \mathbb{R}_{>0}$ the number of layers (here 24), $\vec{\mathbf{H}} \in \mathbb{R}^{d_B \times d_S \times d_H \times d_L}$ the hidden states of a batch, then the following applies

$$\vec{\mathbf{H}}_i = \frac{\sum_{d_S} \left(\vec{\mathbf{H}} \odot (\vec{\mathbf{M}}_i \vec{\mathbf{1}}_{d_L}) \right)}{\sum_{d_S} (\vec{\mathbf{T}}_i \vec{\mathbf{1}}_{\mathbf{d}_L})} \tag{5}$$

In equation 5, we first mask the hidden states $\vec{\mathbf{H}}$ and then add them up over the entire sequence length d_S. We then divide the sum by the number of all positions at which the segment is located and thus obtain the average of all hidden states from the same segment $\vec{\mathbf{H}}$. Figure 2 illustrates a segment-based XAI method in aggregating hidden states of the transformer model. Let's assume that the movie title is "SomeTitle" and will be divided into two parts in the vector representation: "Some##" and "##Title". To summarize and examine the internal vector representation of all positions related to the title, we filter the segments and calculate the average using the masked output embeddings of the transformer.

5. Experiments

5.1. Goal

We will prove that there is a systematic bias in the MovieLens dataset, namely that the distribution of the incoming edges of all movie nodes is subject to strong fluctuations. We use XAI methods to demonstrate that the GraphPrompter model systematically processes the systematic bias and follows a subjectively reasonable processing strategy.

5.2. Set-Up

5.2.1. Hyper Parameters and Hardware

For GraphSage, we adopt the architecture of the PyG tutorial, matching the layer size to GPT2-medium's hidden size 1024 on the output layer and thus also increasing the input layer size to 256. For the linked-neighborhood sampler we keep 20 neighbors in the first hop and 10 neighbors in the second hop. For training the GNN, we increase the batch size to 25600 in order to speed up training process The models were each trained for 2 epochs. We chose the largest possible batch size for the LLM so that it can still be trained on a single GPU which was 128. Training and forwarding took place on an Nvidia A100 SXM4 40 GB and lasted approximately four days in total.

5.2.2. Dataset

The authors of the MovieLens dataset [12] collected movie ratings over a publicly available website[5]. MovieLens 32M includes 32 million movie ratings from 87,585 movies and 200,948 independent users collected in 2023. Each user is identified by a random (user) ID. Each movie is represented by an arbitrary (movie) ID, a title with year of release and a list of genres and tags. The users added tags and ratings to the movies, resulting in facts of the form *(user ID, movie ID, rating, timestamp)*.

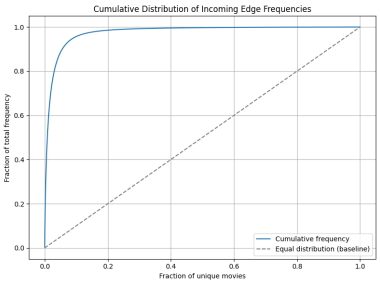
5.2.3. Data Preprocessing

The preprocessing is largely based on the steps in the PyG Link Prediction Tutorial. First, new sequential IDs are assigned to the users and movies. The genres are then one-hot encoded and transferred together with the user and movie IDs and the labels *has rated* into a heterogeneous graph dataset. This is then split 70-15-15 into training, test and validation and the test and validation splits are uniformly assigned negative samples of *not rated* in a 1:1 ratio. In the PyG tutorial, 70% of the training data is only separated for message passing. We refrain from this separation so that we can perform a uniform mapping of the data set into the NLP data set.
The NLP data set is made up of the newly assigned IDs, the movie titles and genres in a fixed structure. To enable the Transformer model to make decisions with or without KGEs, we introduce a special placeholder token *<user KGE>* and *<movie KGE>*, as well as a separator token *<SEP>* for separation. This results in the schema:
<user_KGE><SEP>user ID<SEP><movie_KGE><SEP>movie ID<SEP>movie title<SEP>[movie genres]<SEP>.
We create the segments for each of these data points. We must note that not every feature has the same length in encoded form. During tokenization, we also create the corresponding segments. The segments that we want to keep apart are the *separator tokens <SEP>*, those of the KGEs, the user ID and the NLP movie features *movie-ID*, *title* and *genres*. The NLP dataset is then compared with the graph dataset and split accordingly.

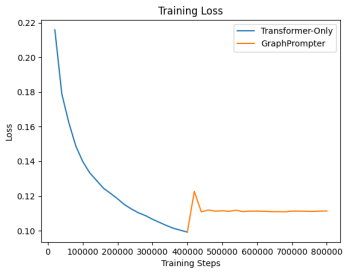
5.2.4. Training

First, we train the GraphSage model and the transformer-Only model for two epochs and evaluate their accuracy against the test dataset. Since our GraphSage model is trained as a regression problem,

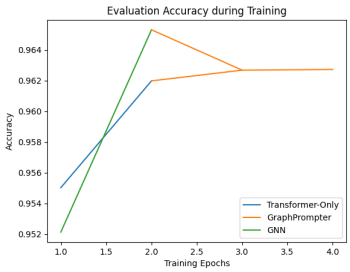
[5]Permalink: https://grouplens.org/datasets/movielens/32m/



(a) The comulative distribution of incoming (positive) edges on the set of all movies



(b) Training losses of Transformer-Only and GraphPrompter models



(c) Training Accuracy on the test dataset after every epoch of training GNN, Transformer-Only and GraphPrompter

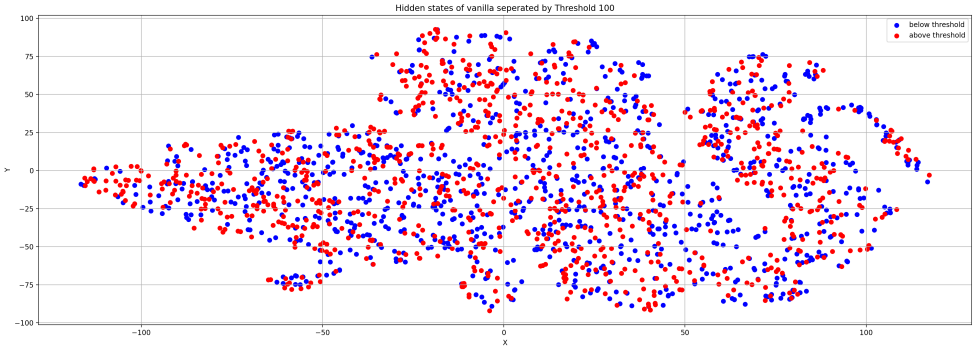
we assume discrete thresholds that best map the regression to the classification. Finally, we train and evaluate the Graphprompter model based on the pre-trained GraphSage and Transformer-only model over two epochs. From the training process, we map the loss and the accuracy on a graph.

5.3. Results

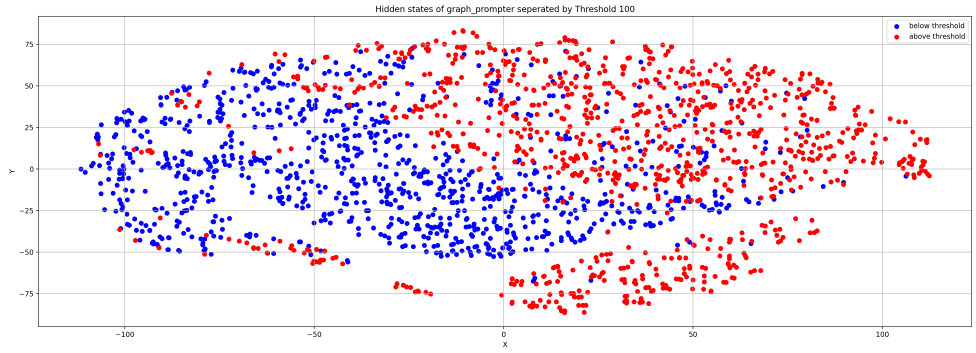
Figure 3a confirms that the majority of all ratings were provided for only a few percent of the total number of movies. In contrast, the distribution of all negative edges is uniformly distributed by design. Figure 3b shows that the loss in the transformer-only model decreases continuously and then finds its minimum at around 0.1. The loss then increases briefly when training the GraphPrompter model and then stabilizes at around 0.11. Figure 3c shows that the GNN's accuracy has the highest performance at around 0.965. The Transformer-Only model achieves the lowest Accuracy of approx. 0.962. The Accuracy of the GraphPrompter model improves by about 0.001 to 0.963 compared to the Transformer-Only model.
Figures 4 show the distribution of the t-sne dimension-reduced hidden states of the movie KGE position in the output layer of both models. Figure 4a shows the distribution of the transformer-only model. As expected, this shows no comprehensible separation. Figure 4b shows the separation of the hidden states of the movie KGE in the output layer of the GraphPrompter. This figure shows a clear (non-linear) separation between movies with many incoming edges and movies with few incoming edges. We conclude that the GraphPrompter model systematically incorporates the underlying bias in the dataset into its decision making process.

6. Discussion

The experiments carried out in this paper show that there is a systematic bias in the MovieLens dataset [12]. A few movies hold a majority of the ratings. This bias was established in our GraphPrompter architecture [7] and can be clearly distinguished in its internal vector representation from the KGEs passed in soft prompts. We interpret this separation as a sign that the model is able to interpret the underlying bias in a systematic way. We can assume that this distinction can help the model to decide between the features produced in the GNN and those produced in the transformer. We assume that the GNN learns (global) neighborhood structures during training with neighborhood Sampling. We also assume that the transformer tends to learn local contexts. We further assume that the Transformer-Only model has a harder time deciding on movies with high input degree, since a global view is needed to differentiate movies with high input degree. In the GraphPrompter model, we combine the strategies of the local and global view of both model architectures. It is possible for the GraphPrompter model to refer to the features generated in the GNN if it makes the decision for a movie with a high input degree and to the transformer-generated features when making decision for a movie with low input degree.
If the GraphPrompter model actually learns such a strategy, then this architecture may lend itself to the



(a) Hidden states of transformer-only model



(b) Hidden states of GraphPrompter

Figure 4: Hidden states of transformer-only and GraphPrompter model's movie KGE position in the last layer reduced dimensionality with t-sne separated by threshold of 100

use of a bias mitigation strategy or other fairness strategies. The discrimination in the model can also be used to favor underrepresented movies, e.g. to promote cultural diversity.

7. Limitations

Figure 4 could be misinterpreted as indicating a lack of structured processing of bias. However, we have only demonstrated that this processing is unlikely to occur in this particular segment. Comparatively structured processing could still occur in other segments of the transformer. We can therefore only conclude that certain properties are present within the GraphPrompter model.

The behaviors of GNN, transformer and GraphPrompter model assumed in this work still have many gaps. The assumption that our transformer learns fewer global structures is based solely on the assumption that it only ever sees one data point at each time during training. Though, the transformer in this experiment is significantly more powerful than the GNN. Nevertheless, an investigation of the actual scopes of all models is appropriate.

The question remains whether the structured processing of an underlying bias in the dataset is more appropriate for fair processing. We assume that such a crucial bias will inevitably have an impact on models trained on it and that structured processing tends to make the system more fair. However, for a more thorough discussion of the fairness of this GraphPrompter implementation, more concrete fairness criteria are needed.

8. Conclusion

The systematic bias in the MovieLens dataset prevails in the internal vector representation of our GraphPrompter model. We argue that the structured processing of this bias rather indicates behavior that increases group fairness. However, further experiments and more concrete fairness criteria are needed to evaluate the practical implications of this perspective.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (GPT-4o) for generating small chunks of Python code, which were adjusted and used in the experiments. Further, the author(s) used DeepL for the translation from German into English. After using these services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] Llama Team, AI @ Meta, The llama 3 herd of models, AI @ Meta (2024). URL: https://ai.meta.com/research/publications/the-llama-3-herd-of-models/.

[2] OpenAI, Gpt-4 technical report, OpenAI (2023). URL: https://cdn.openai.com/papers/gpt-4.pdf.

[3] Gemma Team, Google DeepMind, Gemma 2: Improving open language models at a practical size, ArXiv preprint arXiv:2408.00118 (2024). URL: https://arxiv.org/pdf/2408.00118.

[4] DeepSeek-AI, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, ArXiv preprint arXiv:2501.12948 (2025). URL: https://github.com/deepseek-ai/DeepSeek-R1.

[5] B. Jin, G. Liu, C. Han, M. Jiang, H. Ji, J. Han, Large language models on graphs: A comprehensive survey, IEEE Transactions on Knowledge and Data Engineering 36 (2024) 8622–8642. doi:10.1109/TKDE.2024.3469578.

[6] L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei, B. Long, Graph neural networks for natural language processing: A survey, Foundations and Trends® in Machine Learning 16 (2023) 119–328. URL: http://dx.doi.org/10.1561/2200000096. doi:10.1561/2200000096.

[7] Z. Liu, X. He, Y. Tian, N. V. Chawla, Can we soft prompt llms for graph learning tasks?, in: Companion Proceedings of the ACM Web Conference 2024, WWW '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 481–484. URL: https://doi.org/10.1145/3589335.3651476. doi:10.1145/3589335.3651476.

[8] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, M. Du, Explainability for large language models: A survey, ACM Trans. Intell. Syst. Technol. 15 (2024). doi:10.1145/3639372.

[9] L. Deck, J. Schoeffer, M. De-Arteaga, N. Kühl, A critical survey on fairness benefits of explainable ai, in: Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, FAccT '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 1579–1595. doi:10.1145/3630106.3658990.

[10] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 1025–1035.

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[12] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (2015). URL: https://doi.org/10.1145/2827872. doi:10.1145/2827872.

[13] A. H. Mohammadkhani, C. Tantithamthavorn, H. Hemmatif, Explaining transformer-based code models: What do they learn? when they do not work?, in: 2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM), 2023, pp. 96–106. doi:10.1109/SCAM59687.2023.00020.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 6000–6010.

[15] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: State-of-the-art natural language processing, in: Q. Liu, D. Schlangen (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.

[16] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[17] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao, J. Yang, J. Yuan, Y. Zhao, Y. Wang, X. Luo, M. Zhang, A comprehensive survey on deep graph representation learning, Neural Networks 173 (2024) 106207. doi:https://doi.org/10.1016/j.neunet.2024.106207.

[18] S. Khoshraftar, A. An, A survey on graph representation learning methods, ACM Trans. Intell. Syst. Technol. 15 (2024). doi:10.1145/3633518.

[19] K. Liang, L. Meng, M. Liu, Y. Liu, W. Tu, S. Wang, S. Zhou, X. Liu, F. Sun, K. He, A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal, IEEE Transactions on Pattern Analysis and Machine Intelligence 46 (2024) 9456–9478. doi:10.1109/TPAMI.2024.3417451.

[20] J. Cao, J. Fang, Z. Meng, S. Liang, Knowledge graph embedding: A survey from the perspective of representation spaces, ACM Comput. Surv. 56 (2024). doi:10.1145/3643806.

[21] R. Sun, X. Cao, Y. Zhao, J. Wan, K. Zhou, F. Zhang, Z. Wang, K. Zheng, Multi-modal knowledge graphs for recommender systems, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1405–1414. doi:10.1145/3340531.3411947.

[22] S. Ji, S. Pan, E. Cambria, P. Marttinen, P. S. Yu, A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Transactions on Neural Networks and Learning Systems 33 (2022) 494–514. doi:10.1109/TNNLS.2021.3070843.

[23] F. Akrami, L. Guo, W. Hu, C. Li, Re-evaluating embedding-based knowledge graph completion methods, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1779–1782. doi:10.1145/3269206.3269266.

[24] X. Han, L. Wang, S. N. Han, C. Chen, N. Crespi, R. Farahbakhsh, Link prediction for new users in social networks, in: 2015 IEEE International Conference on Communications (ICC), 2015, pp. 1250–1255. doi:10.1109/ICC.2015.7248494.

[25] X. Song, J. Lian, H. Huang, M. Wu, H. Jin, X. Xie, Friend recommendations with self-rescaling graph neural networks, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 3909–3919. URL: https://doi.org/10.1145/3534678.3539192. doi:10.1145/3534678.3539192.

[26] F. Akrami, L. Guo, W. Hu, C. Li, Re-evaluating embedding-based knowledge graph completion methods, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1779–1782. doi:10.1145/3269206.3269266.

[27] M. Xu, Y. Yin, A similarity index algorithm for link prediction, in: 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 2017, pp. 1–6. doi:10.1109/ISKE.2017.8258724.

[28] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using Siamese BERT-networks, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods

in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 3982–3992. doi:10.18653/v1/D19-1410.

[29] S. Liang, Knowledge graph embedding based on graph neural network, in: 2023 IEEE 39th International Conference on Data Engineering (ICDE), 2023, pp. 3908–3912. doi:10.1109/ICDE55515.2023.00379.

[30] B. Lester, R. Al-Rfou, N. Constant, The power of scale for parameter-efficient prompt tuning, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 3045–3059. doi:10.18653/v1/2021.emnlp-main.243.

[31] A. Kraft, R. Usbeck, The lifecycle of "facts": A survey of social bias in knowledge graphs, in: Y. He, H. Ji, S. Li, Y. Liu, C.-H. Chang (Eds.), Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online only, 2022, pp. 639–652. doi:10.18653/v1/2022.aacl-main.49.

[32] R. Fernando, I. Norton, P. Dogra, R. Sarnaik, H. Wazir, Z. Ren, N. S. Gunda, A. Mukhopadhyay, M. Lutz, Quantifying bias in agentic large language models: A benchmarking approach, in: 2024 5th Information Communication Technologies Conference (ICTC), 2024, pp. 349–353. doi:10.1109/ICTC61510.2024.10601938.

[33] T. Speith, A review of taxonomies of explainable artificial intelligence (xai) methods, in: Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 2239–2250. doi:10.1145/3531146.3534639.

[34] E. N. Volkov, A. N. Averkin, Local explanations for large language models: a brief review of methods, in: 2024 XXVII International Conference on Soft Computing and Measurements (SCM), 2024, pp. 189–192. doi:10.1109/SCM62608.2024.10554222.

[35] L. van der Maaten, G. E. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (2008) 2579–2605.

[36] S. Barocas, M. Hardt, A. Narayanan, Fairness and Machine Learning: Limitations and Opportunities, MIT Press, 2023.

[37] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel, Fairness through awareness, in: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 214–226. URL: https://doi.org/10.1145/2090236.2090255. doi:10.1145/2090236.2090255.

[38] L. Baresi, C. Criscuolo, C. Ghezzi, Understanding fairness requirements for ml-based software, in: 2023 IEEE 31st International Requirements Engineering Conference (RE), 2023, pp. 341–346. doi:10.1109/RE57278.2023.00046.

[39] Z. Chu, Z. Wang, W. Zhang, Fairness in large language models: A taxonomic survey, SIGKDD Explor. Newsl. 26 (2024) 34–48. URL: https://doi.org/10.1145/3682112.3682117. doi:10.1145/3682112.3682117.

[40] A. Singh, T. Joachims, Fairness of exposure in rankings, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 2219–2228. URL: https://doi.org/10.1145/3219819.3220088. doi:10.1145/3219819.3220088.

[41] Y. Deldjoo, D. Jannach, A. Bellogin, A. Difonzo, D. Zanzonelli, Fairness in recommender systems: research landscape and future directions, User Modeling and User-Adapted Interaction 34 (2023) 59–108. URL: https://doi.org/10.1007/s11257-023-09364-z. doi:10.1007/s11257-023-09364-z.

[42] Y. Wang, W. Ma, M. Zhang, Y. Liu, S. Ma, A survey on the fairness of recommender systems, ACM Trans. Inf. Syst. 41 (2023). URL: https://doi.org/10.1145/3547333. doi:10.1145/3547333.

[43] Q. Dong, S.-S. Xie, W.-J. Li, User-item matching for recommendation fairness, IEEE Access 9 (2021) 130389–130398. doi:10.1109/ACCESS.2021.3113975.

[44] Y. Li, H. Chen, Z. Fu, Y. Ge, Y. Zhang, User-oriented fairness in recommendation, in: Proceedings of

the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 624–632. URL: https://doi.org/10.1145/3442381.3449866. doi:10.1145/3442381.3449866.

[45] R. K. Merton, The matthew effect in science, Science 159 (1968) 56–63. URL: https://www.science.org/doi/abs/10.1126/science.159.3810.56. doi:10.1126/science.159.3810.56. arXiv:https://www.science.org/doi/pdf/10.1126/science.159.3810.56.

[46] Z. Zhu, Y. He, X. Zhao, Y. Zhang, J. Wang, J. Caverlee, Popularity-opportunity bias in collaborative filtering, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, WSDM '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 85–93. URL: https://doi.org/10.1145/3437963.3441820. doi:10.1145/3437963.3441820.