Alleviating Cold-Start Problems in Recommendation through Pseudo-Labelling over Knowledge Graph

Shakuntala Mitra, Taylor Hawks

May 9, 2025

1 Introduction

In this project, we re-implement Togashi et al.'s Knowledge Graph-based Pseudo-labeling (KGPL) framework, which is designed to address the cold-start problem in personalized recommendation systems, using PyTorch as our development framework. KGPL integrates users' implicit feedback with a structured knowledge graph and a graph convolutional network (GCN) backbone to enhance sampling for pseudo-labels. This ensures that missing interactions aren't naively treated as negative feedback. By re-weighing entity relations per user and propagating them through the network, KGPL has achieved improved recommendation performance for cold-start users and items compared to other state-of-the-art knowledge graph-aware recommender models.

Overall, the performance of the PyTorch re-implementation is consistent with the original – it reproduces the expected training dynamics and achieves comparable recommendation accuracy. The co-training strategy and knowledge graph pseudo-labeling proved effective, as evidenced by the matching trends and competitive Recall@K values on validation, cold-start subsets, and the overall test set.

2 Dataset

We used the Last.FM dataset, which includes 1,872 users and 3,846 items, resulting in a total of 21,173 observed user-item interactions. This yields a sparsity level of 0.294%, indicating a highly sparse interaction matrix. The associated knowledge graph constructed for this dataset contains 9,366 entities and 60 relation types, encompassing 15,518 triples (Togashi et al. 2020). This dataset has rich item-level metadata and high sparsity, making it suited for evaluating cold-start recommendation performance.

3 Background & Related Works

3.1 Deep Learning in Recommender Systems

Embedding-based approaches address cold start issues by capturing semantic information from a knowledge graph (KG). In contrast, path-based methods rely on manually crafted connectivity patterns (such as meta-paths or meta-graphs) to determine item relatedness tailored to individual users.

The use of graphs in neural recommender systems began with GNNs being applied to homogeneous user-item graphs. More recently, graphs have been used in more intricate ways to enhance recommendations, whether by enhancing the user-item interaction graph, extending the graph with more complex relations, or increasing the aggregation scope within the GNNs. (Cold Start Survey 2025). KGPL falls under the category of graph extension by leveraging knowledge graphs, offering an information-rich improvement to the typical user-item interaction dataset.

3.2 Cold-Start Problem

When new users or new items are introduced, a recommender system is forced to make personalized suggestions without having sufficient user-item interaction data. This is the **cold-start problem**. For a user cold-start, the system has little to no data for a new user's preferences or likings. For an item cold-start, there are no ratings or history of user interactions.

On a technical level, collaborative filtering and embedding-based methods break down when the user-item interaction matrix contains empty rows or columns. When the interaction matrix contains missing entries or very sparse data, similarity calculations cannot produce reliable values. Interaction data sparsity also exacerbates popularity bias towards popular items, creating a negative loop that causes less-established items to be poorly recommended and struggle to receive interactions.

3.3 Knowledge Graph

Knowledge graphs (KGs) are one tool used to mitigate the cold-start problem, and play a key role in Togashi et al.'s proposed framework. A knowledge graph is a structured, heterogeneous graph representation of entities, such as users, items, or attributes, and the relations between those entities. The KGPL framework leverages the KG to inform pseudo-labeling strategies. The KG is constructed using item metadata. The nodes represent the users, items, and attributes, while the edges represent entity relations such as item-attribute, attribute-attribute, and item-item relationships. For example, in context of the MovieLens1M dataset, the KG provides information about movies (items) in the form of interconnected entities (e.g. actors, directors, genres) and relations. A knowledge graph represents facts as triplets (head, relation, tail), where head and tail are entities and relation is the link between them.

The KG is used in two major ways:

1. Guide pseudo-labeling by selecting additional training samples by finding related items to a user via meta-paths. Meta-paths are sequences of relations that capture meaningful semantic connections between entities.

2. Propagate information through a Graph Neural Network (GNN) that enriches item representations.

3.4 Pseudo-Labeling

Traditional recommendation models generally treat unobserved user-item interactions as negative feedback, which can bias the system against new items and reinforce popularity bias. KGPL considers some of these unobserved interactions as potential positive instances, alleviating some of the popularity bias found in other recommendation models. For each user, the model explores the KG to find items connected to the user's previously interacted items through meta-paths. Items reachable via meta-paths are considered likely to align with the user's preferences, as they share contextual or attribute-based similarities with known positive items. By assigning pseudo-labels to these items, the model effectively augments the training data, enhancing its ability to make accurate recommendations, especially in cold-start scenarios where user-interaction data is sparse.

4 Methods

Problem Formulation. In personalized recommendation scenarios, we consider a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. User-item interactions are represented by a binary matrix $\mathbf{Y} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, where each entry $y_{u,i} = 1$ indicates that user $u \in \mathcal{U}$ has engaged with item $i \in \mathcal{I}$. Due to data sparsity, many entries in $\mathbf{Y}$ are unobserved. For each user u, we define $\mathcal{I}_u^+ = \{i \in \mathcal{I} \mid y_{u,i} = 1\}$ as the set of items with observed interactions, and $\mathcal{I} \setminus \mathcal{I}_u^+$ as the set of unobserved items. The collection of observed user-item interactions is then denoted by $\mathcal{O} = \{(u,i) \mid u \in \mathcal{U}, i \in \mathcal{I}_u^+\}$. To incorporate auxiliary semantic information, we utilize a knowledge graph (KG) defined as $\mathcal{G} = \{(h,r,t) \mid h,t \in \mathcal{E}, r \in \mathcal{R}\}$, where each triplet (h,r,t) represents a relation $r \in \mathcal{R}$ between a head entity h and a tail entity t, both drawn from the entity set $\mathcal{E}$. This entity set includes both items (i.e., $\mathcal{I} \subseteq \mathcal{E}$) and non-item entities ($\mathcal{E} \setminus \mathcal{I}$), such as item attributes or contextual information. The KG provides structured connectivity among entities, enabling the model to reason over multi-hop relational paths for more informed recommendation.

The main goal of this project is to predict $y_{u,i}$ for each unobserved sample $(u,i) \in (\mathcal{U} \times \mathcal{I}) \setminus \mathcal{O}$ based on the observed interactions $y_{u,i}$ corresponding to $\mathcal{O}$. We want to procure a prediction function $\hat{y}_{u,i} = \mathcal{F}(u, i \mid \Theta, Y, \mathcal{G})$, where $\hat{y}_{u,i}$ is the likelihood of a user u engaging with item i, and Θ are model parameters of function $\mathcal{F}$ (Togashi et al., 2020).

4.1 Knowledge Graph Construction

The knowledge graph $\mathcal{G}$ is constructed by using the user-item interaction data. Each item $i \in \mathcal{I}$ is mapped to a corresponding entity $e_i \in \mathcal{E}$ using a mapping file. The KG comprises triplets (h, r, t), where h and t are entities, and r is the relation between them. These triplets are extracted from the knowledge base and represent various relationships.

4.2 Graph Neural Network Architecture

We preserved the model architecture from Togashi et al. (2020)'s KGPL model. The core is a GNN encoder that propagates information over a KG in a user-specific way. For each user u, we construct a personalized adjacency matrix A_u whose (j, k) entry is defined as $(A_u)jk = \exp!\big(u^\top re_j, e_k\big)$, where r_{e_j,e_k} is the embedding of the relation connecting entity e_j to e_k in the KG (and $(A_u)_{jk} = 0$ if no direct relation exists). This weights each KG edge by the affinity between user u's latent preferences and the edge's relation type. We then perform L-layer propagation on the KG. At each layer l, the hidden entity representations H_l are updated as

$$H_{l+1} = \sigma_l(D_u^{-1/2}(A_u + I)D_u^{-1/2} H_l W_l), \text{where } l = 0, 1, ..., L - 1$$

where $H_0 = E$ (trainable initial entity embeddings), I is the identity (self-loop) matrix, D_u is the diagonal degree matrix of $A_u + I$, and W_l are layer-specific weight matrices. We use `LeakyReLU` activations for the intermediate aggregator layers and $tanh$ for the final layer.

4.3 Mini-Batch Augmentation via Pseudo-Labeling

We use a semi-supervised pseudo-labeling approach that augments the sparse observed interactions with inferred labels for unobserved user–item pairs. Let I_u^+ be the set of items that user u has interacted with (ground-truth positives), and I_u^- represent unobserved pairs treated as negatives. This is the labeling function, which assigns a label of 1 to any observed interaction, 0 to a negative instance, and uses the model's own prediction as a "soft" label for certain unobserved pairs.

$$\hat{l}_u(i) = \begin{cases} 1 & \text{if } i \in \mathcal{I}_u^+ \\ \hat{y}_{u,i} & \text{if } i \in \mathcal{I}_u^\pm \\ 0 & \text{if } i \in \mathcal{I}_u^- \end{cases}$$

4.4 Binary Cross-Entropy Loss

The model is optimized by minimizing the classification loss over all three categories of interactions (positive, pseudo-labeled, and negative). The total loss is the sum of the binary cross-entropy loss components per user u:

$$\mathcal{L} = \sum_{u \in \mathcal{U}} \left(\sum_{i \in \mathcal{I}_u^+} \ell(1, \hat{y}_{u,i}) + \sum_{i \in \mathcal{I}_u^\pm} \alpha_{u,i}\, \ell(\hat{l}_u(i), \hat{y}_{u,i}) + \sum_{i \in \mathcal{I}_u^-} \beta_{u,i}\, \ell(0, \hat{y}_{u,i}) \right)$$

where $\alpha_{u,i}$ and $\beta_{u,i}$ are weighting factors which control the skewness for pseudo-positive and negative sampling. This pseudo-labeling strategy provides additional training signals for unobserved pairs without assuming all of them are negative. However, two challenges arise:

summing the loss over all $|U| \times |I|$ user-item pairs is infeasible in large systems, and the pseudo-labels $\hat{y}u, i$ can be noisy since they are model-generated. We address these issues through careful sampling and our co-training approach.

4.5 Knowledge-Graph Aware Sampling of Unobserved Items

The sampling probability for an unobserved item i for user u is defined as:

$$q(i|u) \propto n_{u,i}^{a}$$

where $n_{u,i}$ is the number of paths connecting user u to item i in the KG, and a is a hyperparameter controlling the influence of path count. This approach allows the model to assign pseudo-labels to items that share KG connections to a user's positively interacted items, enhancing the training data and improving recommendation performance in cold-start scenarios.

To counter popularity bias, the negatives are samples in proportion to item popularity:

$$p(i^- \mid u) = \frac{(m_{u,i^-})^b}{\sum_{j \in \mathcal{I} \setminus \mathcal{I}_u^+} (m_{u,j})^b}$$

4.6 Co-Training Approach

We maintain two models, f and g, which are initialized independently but share the same design. In each training iteration, we generate two mini-batches B_f and B_g using the sampling procedure above. Model f is used to assign pseudo-labels for all $(u, i) \in B_f$, and model g provides labels for the pairs in B_g. We then cross-update the models: f is trained on the batch B_g using labels from g, while g is trained on B_f using f's labels.

$$\mathcal{L}_{B_g}(f) = \sum_{(u,i,l_{u,i}^{(g)}) \in B_g} \ell \left(l_{u,i}^{(g)},\ \hat{y}_{u,i}^{(f)};\ \Theta_f \right)$$

where $l_{u,i}^{(g)} = \hat{l}u(i)$ is the pseudo-label determined by model g for user u and item i (using the labeling function of Eq. (3)). After convergence, we discard one model and use the remaining one to provide the recommendations.

5 Results

For summary tables of training and evaluation metrics, as well as relevant figures, please see the Appendix.

Over 40 epochs of co-training, both student models (f and g) exhibited steadily decreasing training loss and closely matched progress. Model f's average training loss dropped from 5.04 in the first epoch to 1.81 by the 40th epoch, with model g following an almost identical trajectory. This indicates stable co-training, without one network dominating. The significant loss reduction suggests that the networks successfully learned from both observed and pseudo-labeled instances. The co-training process converged to similar performance levels, reflecting the intended robustness of this approach. Recommendation accuracy on the validation set improved significantly over training. Model f's Recall@20 climbed from only 0.67% after the first epoch to 15.3% by epoch 40. Recall@5 and Recall@10 reached approximately 6.2% and 9.4%, respectively, in the final epoch. The most rapid gains occurred in the early epochs; for example, Recall@20 was 0.081 by epoch 19. This shows the model learned the majority of useful signal in the first half of training, then gradually fine-tuned in later epochs. Overall, the increasing validation Recall@K across epochs demonstrates that the PyTorch implementation achieves the expected learning behavior of KGPL, with no signs of overfitting (validation metrics plateau rather than decline toward the end).

5.1 Cold-Start Users

For users with $\leq$1 prior interaction, the model's recommendation quality was very limited: Recall@20 was only about 8.3%, and Recall@5/10 were effectively 0%. However, for users with $\leq$2 interactions, Recall@20 jumped to 20.3%, and performance continued to improve as we allowed more historical interactions. For users with up to 5 or 10 interactions, the model's recall approaches that of the overall user population. This upward trend demonstrates that our KGPL model partially alleviates cold-start limitations. In summary, the PyTorch model behaves consistently with expectations – struggling on entirely new users, but gaining accuracy as more data becomes available – confirming that the pseudo-labeling over the knowledge graph provides measurable benefit in cold-start scenarios.

5.2 Top-K Evaluation

On the full test set, model f achieved a Recall@5 of 7.1%, Recall@10 of 12.4%, and Recall@20 of 17.6%. In comparison, the original TensorFlow-based implementation reported slightly higher top-K recall values (e.g. Recall@5 $\approx$ 9.93%, Recall@10 $\approx$ 15.47%, Recall@20 $\approx$ 22.25%). The PyTorch model's metrics are of the same order of magnitude and follow the same pattern, albeit falling a few percentage points short in recall. This small difference might be due to minor implementation or environment setup differences. The Precision@K metrics show a similar alignment: for instance, the PyTorch model attains 2.0% Precision@20, compared to 2.3% in the TensorFlow output logs. Overall, the performance of the PyTorch KGPL reimplementation is consistent with the original, meaning that our re-implementation was successful.

6 Conclusions & Future Work

One promising direction for future work is to shift the learned adjacency mechanism in KGPL from the user level to the item level. Rather than learning user-specific relation weights, this approach would focus on capturing how entity relations influence the semantics of each individual item. We believe that an item's relevance is more richly represented by entity relations that reflect a user's taste toward that specific item, rather than by relations aggregated across all of a user's activities. As such, we hypothesize that modeling an item's relevance through its local relational context—independent of aggregated user behavior—may yield more fine-grained and transferable embeddings. These item-level representations could then be re-aggregated per user to estimate engagement likelihoods. Incorporating this item-focused relational learning into KGPL's existing pseudo-labeling framework may further enhance recommendation accuracy in cold-start settings.

7 References

Togashi, R., Otani, M., & Shin'ichi Satoh. (2020). Alleviating Cold-Start Problems in Recommendation through Pseudo-Labelling over Knowledge Graph. *ArXiv (Cornell University).* https://doi.org/10.48550/arxiv.2011.05061

Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., & Wang, Z. (2019). Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. *ArXiv (Cornell University).* https://doi.org/10.48550/arxiv.1905.04413

Zhao, X., Wang, M., Zhao, X., Li, J., Zhou, S., Yin, D., Li, Q., Tang, J., & Guo, R. (2023). Embedding in Recommender Systems: A Survey. *ArXiv* https://doi.org/10.48550/arXiv.2310.18608

Zhang, W., Bei, Y., Yang, L., Zou, H. P., Zhou, P., Liu, A., Li, Y., Chen, H., Wang, J., Wang, Y., Huang, F., Zhou, S., Bu, J., Lin, A., Caverlee, J., Karray, F., King, I., & Yu, P. S. (2025). Cold-Start Recommendation towards the Era of Large Language Models (LLMs): A Comprehensive Survey and Roadmap. *ArXiv* https://doi.org/10.48550/arXiv.2501.01945

8 Appendix

8.1 Tables and Figures

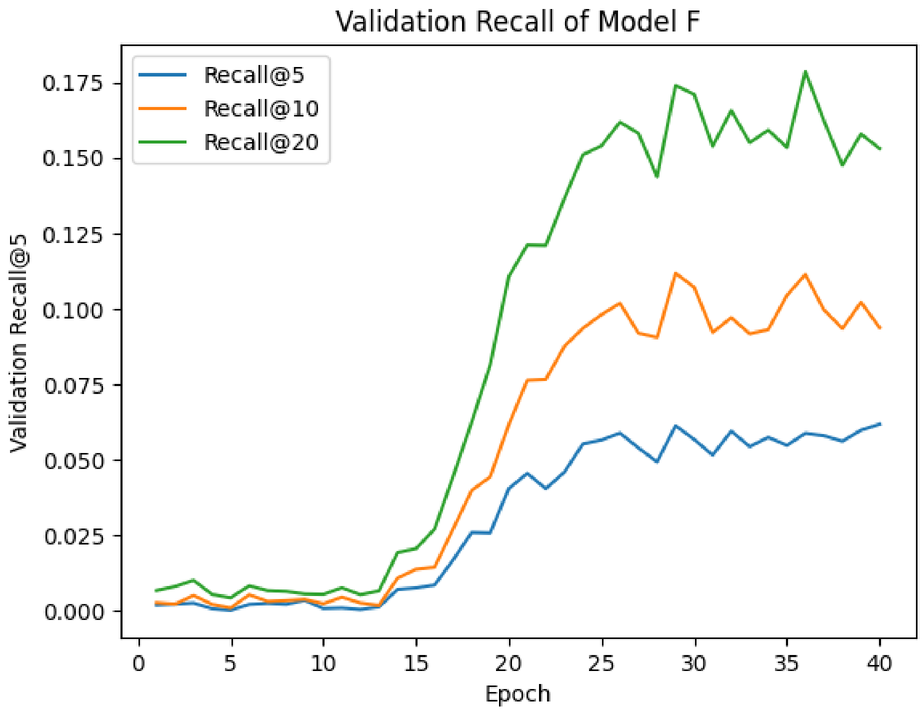

Figure 1: Validation recall of Model F across multiple recall cutoffs, over 40 epochs of co-training.

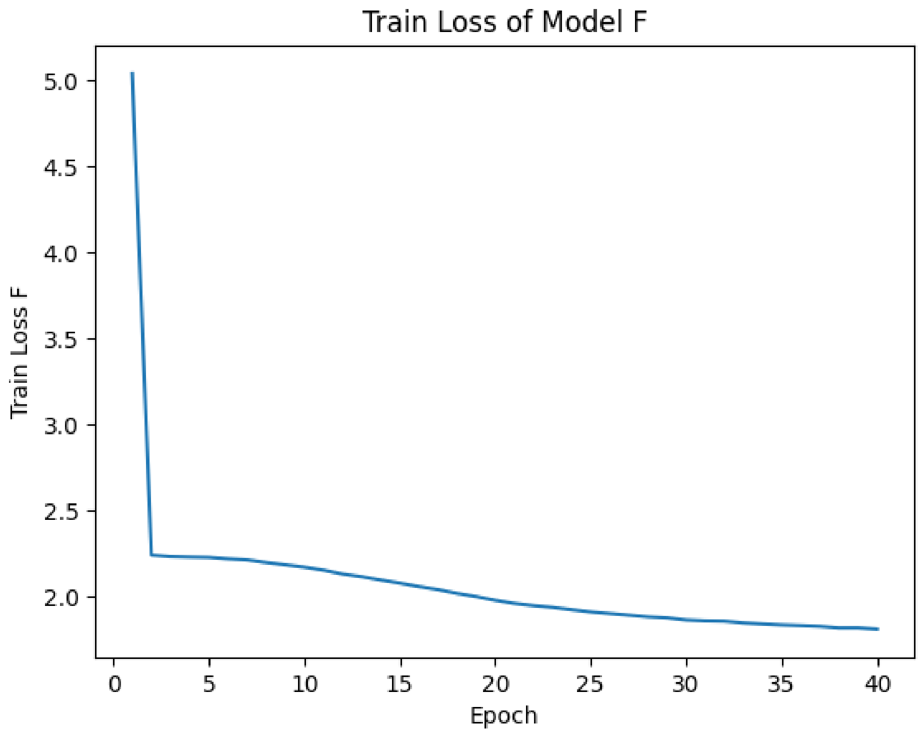

Figure 2: Training loss of Model F over 40 epochs of co-training.

Cold Start	**P@1**	**R@1**	**P@2**	**R@2**	**P@5**	**R@5**	**P@10**	**R@10**	**P@20**	**R@20**
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0056	0.0833
2	0.0656	0.0230	0.0574	0.0434	0.0557	0.1046	0.0361	0.1311	0.0238	0.2033
3	0.0426	0.0170	0.0390	0.0330	0.0383	0.0717	0.0284	0.1128	0.0216	0.1966
4	0.0374	0.0144	0.0459	0.0457	0.0422	0.0902	0.0320	0.1389	0.0245	0.2185
5	0.0446	0.0172	0.0456	0.0415	0.0389	0.0787	0.0320	0.1296	0.0245	0.2046
6	0.0482	0.0207	0.0439	0.0411	0.0371	0.0764	0.0313	0.1281	0.0238	0.1965
7	0.0531	0.0232	0.0456	0.0448	0.0388	0.0822	0.0311	0.1288	0.0238	0.1995
8	0.0480	0.0211	0.0450	0.0441	0.0386	0.0831	0.0304	0.1290	0.0233	0.1958
9	0.0510	0.0245	0.0450	0.0460	0.0372	0.0823	0.0302	0.1310	0.0226	0.1944
10	0.0390	0.0191	0.0395	0.0413	0.0334	0.0767	0.0281	0.1265	0.0214	0.1886

Table 1: Cold-start recommendation performance across multiple precision and recall cutoffs (rounded to 4 decimal places).

Top-K	**Precision**	**Recall**
1	0.0390	0.0191
2	0.0360	0.0363
5	0.0304	0.0707
10	0.0279	0.1236
20	0.0203	0.1762

Table 2: Overall Top-K recommendation performance on the test set (rounded to 4 decimal places).