The Impact of Graph Structure, Cluster Centroid and Text Review Embeddings on Recommendation Methods

Dolog, Peter; Torres, Sergio David Rico; Velaj, Yllka; Sadikaj, Ylli; Stephan, Andreas; Roth, Benjamin; Plant, Claudia

Published in:
ACM Transactions on Recommender Systems

DOI (link to publication from Publisher):
10.1145/3765630

Publication date:
2026

Document Version
Accepted author manuscript, peer reviewed version

Citation for published version (APA):
Dolog, P., Torres, S. D. R., Velaj, Y., Sadikaj, Y., Stephan, A., Roth, B., & Plant, C. (2026). The Impact of Graph Structure, Cluster Centroid and Text Review Embeddings on Recommendation Methods. *ACM Transactions on Recommender Systems*, *4*(2), Article 24. Advance online publication. https://doi.org/10.1145/3765630

The Impact of Graph Structure, Cluster Centroid and Text Review Embeddings on Recommendation Methods

PETER DOLOG, Department of Computer Science, Aalborg University, Denmark
SERGIO DAVID RICO TORRES, Faculty of Computer Science, University of Vienna, Austria
YLLKA VELAJ, Faculty of Computer Science and ds:Univie, University of Vienna, Austria
YLLI SADIKAJ, Faculty of Computer Science and UniVie Doctoral School Computer Science, University of Vienna, Austria
ANDREAS STEPHAN, Faculty of Computer Science and UniVie Doctoral School Computer Science, University of Vienna, Austria
BENJAMIN ROTH, Faculty of Computer Science and Faculty of Philological and Cultural Studies, University of Vienna, Austria
CLAUDIA PLANT, Faculty of Computer Science and ds:Univie, University of Vienna, Austria

It is generally accepted that collaborative information is important for the performance of recommender systems. It is also generally accepted that if this information is sparser, it impacts recommendation systems negatively. Various approaches have tried to lift this problem by employing side information. However, global patterns that can be provided by clusters of similar items and users or even additional information such as text are often not used together with collaborative information. We study the impact of integrating clustering embeddings, review embeddings, and their combinations with embeddings obtained by a recommender system. We study the performance of this approach across various state-of-the-art recommender system algorithms including graph-based methods. We highlight that graph structures are important with sparser datasets and both, in knowledge graphs with side information as well as in collaborative bipartite graphs. In less sparse datasets, a collaborative bipartite graph is usually sufficient. We also highlight that the improvement of recommendation performance through clustering, particularly evident when combined with review embeddings is most visible on sparser data, while on less sparse data incorporating review embeddings may be sufficient when combined with one of the graph-based methods, or otherwise when combined with clustering in other methods.

CCS Concepts: • **Information systems** → **Recommender systems**; **Clustering**; • **Computing methodologies** → *Information extraction.*

Additional Key Words and Phrases: Clustering, Graph-Based Methods, Text Embedding, Recommender Systems

Authors' addresses: Peter Dolog, Department of Computer Science, Aalborg University, Selma Lagerlöfs Vej 300, DK-9220, Aalborg–East, Denmark, dolog@cs.aau.dk; Sergio David Rico Torres, Faculty of Computer Science, and University of Vienna, Vienna, Austria, ricotorres01@unet.univie.ac.at; Yllka Velaj, Faculty of Computer Science and ds:Univie, and University of Vienna, Vienna, Austria, yllka.velaj@univie.ac.at; Ylli Sadikaj, Faculty of Computer Science and UniVie Doctoral School Computer Science, University of Vienna, Vienna, Austria, ylli.sadikaj@univie.ac.at; Andreas Stephan, Faculty of Computer Science and UniVie Doctoral School Computer Science, University of Vienna, Vienna, Austria, andreas.stephan@univie.ac.at; Benjamin Roth, Faculty of Computer Science and Faculty of Philological and Cultural Studies, University of Vienna, Vienna, Austria, benjamin.roth@univie.ac.at; Claudia Plant, Faculty of Computer Science and ds:Univie, and University of Vienna, Vienna, Austria, claudia.plant@univie.ac.at.

ACM 2770-6699/2025/0-ART0
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Peter Dolog, Sergio David Rico Torres, Yllka Velaj, Ylli Sadikaj, Andreas Stephan, Benjamin Roth, and Claudia Plant. 2025. The Impact of Graph Structure, Cluster Centroid and Text Review Embeddings on Recommendation Methods. *ACM Trans. Recomm. Syst.* 0, 0, Article 0 (2025), 28 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recommender systems try to find items from the recommendation domain that are the most relevant to user interests or preferences. The key is to map the interests or preferences to a common space with items where a similarity measure can be used to identify those items that are closer to a user's preference or interest. Each item and user is typically represented as an embedding vector in such a common space. Items closer to the preference or interest in such space are then recommended to the user.

Different approaches exist in the literature ranging from for example k-nearest-neighbor methods [38], through dimensionality reduction methods [37], and different neural network architectures [17] to reduce the dimensionality of items and users embedding vectors. Furthermore, graph-based methods have been recently proposed, considering for example knowledge graphs [51], which connect entities representing knowledge about items expanded with edges between users and items representing rating behavior. Techniques that use graphs promise that by exploiting relations between items and users, they would improve recommendation performance over the methods using less structured data.

It is widely accepted that user rating history is a good source of information for learning a user's preferences and interests. In addition, review data is recently considered a good data source for preference and interest learning. The review data seems to serve as a good complement for missing information about user ratings. We hypothesize that the insights gained from analyzing review text may be even more robust or enhance the patterns observed in rating behaviour.

The main motivation for using reviews is that they provide explicit evidence on sentiments on aspects of the items reviewed. When users write a review, we tend to believe that the user is really interested (be it positively or negatively) in the item he/she is writing about. Further, the text often contains mentions and sentiments for specific features of such an item. Therefore, learning features from the review text provide additional information which sometimes reflects even better the preference on certain aspects of the items than standard metadata.

Furthermore, similar to the idea of finding closely related items to a user's preference in a common vector space, clustering can be employed to group relevant data points in such a common space. Clustering algorithms often operate on embedded data points into a common vector space where a similarity measure is used to group together relevant data points. The difference is that clustering algorithms are mostly unsupervised, [40], while recommender system methods require labeled data.

We are motivated by the fact that the grouped users or items have common characteristics. Centroids of such groups provide a prototype for a representation of items or users in the groups. Adding this information into collaborative and/or review representation provides an opportunity to better distinguish between interesting (preferred) and not interesting (not preferred) items since centroid features make similar items or users even closer to each other. We further hypothesize that such features are helping to bring additional items similar to centroid representations higher on the recommendation lists.

While there are works which have studied such aspects for particular methods proposals and in isolation, to the best of our knowledge, a larger study, where such additional sources of features are used together on state-of-the-art methods to clarify their impact, is not available.

In this paper, we study how graph structure, clustering, review embeddings, and their combinations influence recommender systems. To this end, we provide the following contributions:

- We propose a methodology for combining clustering and review embeddings with embeddings obtained from a recommender system. We study the performance of this methodology on different representative state-of-the-art recommender system algorithms.
- A graph clustering algorithm, SpectralMix [40] is adapted, for the first time, for recommendations with two additional variants of objective functions: DOT product and multiplication between DOT product and Euclidean Distance.

Thanks to our comprehensive experimental evaluation, we gain the following insights:

- Integrating review embeddings into embeddings obtained by all methods improves the recommendation performance. The improvement on Mindreader is larger than the improvement on the Amazon Book or Yelp dataset.
- Methods which integrate all aspects considered in this paper perform the best on sparser datasets such as Amazon Book.
- Graph-based methods play an important role and have good performance. Both knowledge graphs, as well as collaborative graphs, provide a good structure for learning models for recommendation.
- Embeddings from clustering alone improve the recommendation performance on embeddings obtained from non-neural methods.
- Embeddings from clustering alone does not usually help on most graph based neural methods when considered without review embeddings.
- Side information with structure is an important aid in sparser datasets.
- Evaluation methodology and databases split influences the results.

This paper is an extended version of a short paper published previously at The Web Conference [9]. In this paper, we provide more details on the methods and methodology. it is also extended with additional experimental results on the Amazon Book dataset and the Yelp dataset.

The rest of the paper is structured as follows. Section 3 discusses related work. Section 3 describes the methods we investigate and the methodology of how we integrate various components together. Section 4 provides insight into the experimental setup and experimental results obtained from this work. Section 5 discusses conclusions and future work.

2 RELATED WORK

Existing related work has primarily focused on individual aspects of enhancing recommender systems, such as integrating user reviews, incorporating clustering techniques, or leveraging graph-based representations. Each of these dimensions has been explored in isolation, with dedicated surveys addressing them separately. For instance, graph-based recommendation approaches are extensively reviewed in [11], while review-based systems are covered in [14]. However, to our knowledge, there is no comprehensive survey focusing specifically on clustering-based recommendation methods. In this work, we aim to fill this gap by examining the combined impact of three key components, i.e., graph structure, review text, and clustering, on the performance of recommender systems. We investigate how each of these elements contributes to recommendation quality, both independently and in combination. To the best of our knowledge, this integrated perspective is novel and has not been explored before.

Clustering. When examining clustering independently within recommendation research, recent methods have increasingly focused on the adoption of clustering to better understand user intent in recommendation scenarios. The method proposed in [26] models user intents as learnable clusters and provides a method which simultaneously optimizes recommendation as well as clustering. A combination of contrastive learning and K-Means clustering for intent understanding is studied in [6]. The authors of [28] coin intention clustering in connection to disentangled sequence encoding. They try to predict future sequences fitting also different intentions and use clustering with respect to intention prototypes. Another work, [35], builds on top of it and further refines the clustering method by a fine-grain intent contrastive leaning so that the predicted subsequences get even closer to the intention prototypes. Differently from intent clustering and discovery, [31] uses spectral clustering for capturing collaborative signals in user and item groups in the context of transformer-based generative recommendation models. This perspective on clustering aligns with how we incorporate clustering into our analysis of recommender systems in this study. In [32], K-Means clustering is utilized for grouping semantically very related tags in attribute selection for the recommender system. Other works, such as [21] provide insights into the scalability and performance benefits of clustering for social tagging data in recommenders. [41] studies the use of the context-dependent variant of hierarchical clustering in recommender. In [43] clustering is used to obtain a multi-faceted grouping of users for finding their prototypical interests. In [44], authors propose to use clustering for recommendation such that they recommend items a user has not seen yet and purchased by users from the same cluster to which a user is assigned. Several clustering algorithms are compared there, but the impact on the state-of-the-art recommendation methods is not studied. A multi-clustering approach is used in a fashion recommender system with neighborhood approaches [1]. Clustering has been applied with various objectives in recommender systems, often to improve the performance for specific goals. In contrast, we aim to investigate how clustering, both independently and in combination with review data, affects a set of recommendation methods, including those not originally designed with clustering in mind. We also explore the feasibility of extending these methods with clustering in a straightforward and generalizable manner.

Review text. Review text has been considered in hybrid recommenders. The following works [4, 14] give an overview of the approaches in the literature. It is recognized that review text may help recommendation systems to handle data sparsity by adding additional reference sources, and by doing so, improving accuracy [14]. In addition, it may help in the explanation process for recommendation algorithms as well [14]. For example, recent works on review aware recommendation [48] employ contrastive learning in both review text space and collaborative space, and show improvements over selected baselines. The method proposed in [24] uses dual attention in convolutional networks both for rating and review data to address the cold start problem. Set encoders and attentions are used together with three distinct views, collaborative (graph), short-term sequence, and a set view for long-term preference to capture multiple aspects of user preferences. The authors in [50] use BERT to capture user behaviour and preferences for user modelling in two contrastive learning sub-tasks. Very recent works, as [27], propose a recommender system based on prompting of Large Language Models (LLM) and going beyond using review text. Additionally, [25] improves content-based recommender with LLMs. Text embedding methods have been used in recommenders where the mean squared error is used to measure a performance [8]. [47] reported on using categorical, title, and text descriptions of items as a side information beyond review text in their embedding methods. In our work, we focus on how the review text component can affect any kind of recommendation system when it is added in a simple way such as concatenation to learned embeddings from other preference sources, i.e., user rating behaviour or connected entities in a knowledge graph. We aim to analyze the impact this addition has and whether already simple methods would be practically more viable when adding clustering and review text into their recommendation process.

Graphs. The authors of [11] provide a comprehensive survey of graph-based recommender systems. Major work has been devoted to utilizing graph-based methods and graph-based neural networks to further improve collaborative recommendation systems. CGCL [16] and AdaGCL [19] are examples of that. Both use a form of contrastive learning. While CGCL uses LightGCN as a backbone and adds contrastive learning with candidate and neighbour user/item on top of it, AdaGCL combines BPR loss with contrastive learning on a graph generative model and a graph denoising model as view generators. Typically, convolutions and attentions are mechanisms used on graph models for recommendations. Simplified graph convolutions are used for example in LightGCN [17] or in [12]. GraphSage [13] uses a message passing and walk-based learning of node embeddings. Graph attentions are used in Fi-GNN [23]. Different from collaborative filtering, KGCL [51] uses contrastive learning on their knowledge graph, a heterogeneous graph of entities and relations between them formed from metadata on items. KGCL also utilizes LightGCN backbone for the collaborative source of preference, but in addition, it also uses KGAT [46], an attention-based graph network to capture most relevant relations and entities from the knowledge graph. In our work, we investigate how, not only, methods which combine graphs for a collaborative signal with review text, as well as clustering, but also methods based on learning from knowledge graphs, are impacted by clustering and review text embeddings when they are used in recommendation scenarios.

Our Study. To summarize, in our work, we study in detail the impact of clustering, review text embeddings, and connected entities in a knowledge graph on recommendation. We comment on the role of graph structure either in the knowledge graph or in a collaborative bipartite graph. We use sentence embeddings instead of word embeddings to investigate the impact of the text reviews. We focus our study on simple text review embedding aggregations for users and items to show the impact on recommendation accuracy, but further processing is possible. In this paper, we provide a unique analysis of the impact of clustering, review text embeddings, graph structure, and their combination on different selected state-of-the-art methods for recommendation systems. We choose representative methods from the neighbourhood-based class, collaborative filtering class with dimensionality reduction, graph convolutions, graph-based methods considering contrastive learning, and spectral analysis considering homogeneous and heterogeneous graphs. We extend our previous work [9] with more details on the methodology, and we provide additional experiments on the Amazon Book and Yelp datasets. We also touch upon the impact of graphs with side information and relation to the density of the ratings on datasets and their impact on the methods.

3 METHODOLOGY

Our goal is to show the impact of graph structures, clustering, and reviews on recommendation methods. To this end, we identify representative methods on which we would like to study the effects and describe their principles. Further, we describe how learning and prediction work in general with baseline methods. Finally, we describe the ways we integrate cluster centroids and review embeddings in predictions.

3.1 Methods Selection

We select the following representative methods, including statistical, deep learning, and spectral clustering-based approaches: Knowledge Graph Contrastive Learning (KGCL) for recommendation [51], LightGCN [17], Candidate-aware Graph Contrastive Learning for Recommendation (CGCL) [16], Adaptive Graph Contrastive Learning for Recommendation (AdaGCL) [19], Bayesian Personalized Ranking based Matrix Factorization (BPRMF) [37], K-Nearest Neighbor (KNN) [38], and a matrix factorization method with alternate least square optimization [18]. In addition, we select two spectral clustering methods: a traditional spectral clustering [45] and the SpectralMix [40], which work on graphs with multiple entity and relation types as representative graph clustering algorithms.

BPRMF. BPRMF is a representative matrix factorization method that produces user and item vectors with significantly reduced dimensionality. It learns the most representative features of items and user profiles from implicit rating data. Using the following loss function, it estimates the probability that a user u prefers item i ($\hat{y}_{ui}$) over item j ($\hat{y}_{uj}$) as follows [37]:

$$\mathcal{L}_{BPR} = \sum_{(u,i,j)\in D} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) - \lambda_{\Theta}||\Theta||^2 \tag{1}$$

where D is a training set, σ is a logistic sigmoid function, λ_{Θ} is a parameter determining to which extent the regularization parameters are considered, and Θ are regularization parameters.

LightGCN. LightGCN is a collaborative filtering method where user and item embeddings are computed in convolutions over a collaborative rating graph. It belongs to a group of neural graph convolutional techniques with a minimal set of components in its neural network architecture. In various experiments, it performed better than more complex neural collaborative filtering algorithms. The specific aggregation mechanism for convolutions for the next layer ($k + 1$) of neighbours can be described as follows [17]:

$$\mathbf{e}_u^{(k+1)} = AGG(\mathbf{e}_u^{(k)}, \mathbf{e}_i^{(k)} : i \in \mathcal{N}_u) \tag{2}$$

where $\mathbf{e}_X$ are embeddings of users or items, $\mathcal{N}_u$ are user neighbors, and $AGG(\cdot)$ is an aggregation function. By doing so, the embeddings of users and items aggregate signals from their neighbours. LightGCN uses the BPR loss function to estimate a preference of user u for item i over item j [17] (see Eq. 1).

KGCL. KGCL is a Knowledge Graph (KG) based method with a contrastive learning component bringing a de-noising component that overcomes noise and sparsity in knowledge graphs to improve preference capturing.

KGCL features three main components: Relation-aware Knowledge Aggregation, Knowledge Graph Augmentation, and Knowledge-Guided Contrastive Learning.

Relation-aware Knowledge Aggregation serves as a component that identifies through an attention mechanism which relations between an item and neighbouring entities are relevant for the current item and aggregates their scores into the current item embeddings as follows:

$$\mathbf{x_i} = \mathbf{x_i} + \sum_{e\in\mathcal{N}_i} \alpha(e, r_{e,i}, i)\mathbf{x_e} \tag{3}$$

where $\mathcal{N}_i$ are entities which are item i neighbors through different relation types $r_{e,i}$, and $\mathbf{x_i}$ and $\mathbf{x_e}$ are item and entity embeddings respectively. Here, α is an attention function that selects relevant entities based on a learned weight matrix and *leakyReLU* [51]. As an alternative, to further enhance the semantic multi-relational representation, KGCL uses the TransE method [2].

Knowledge Graph Augmentation produces auxiliary self-supervised signals for the contrastive learning component. It generates two correlated data views over a knowledge graph which select a subset of triplets based on masking vectors with probabilities for selecting the triplets. The views provide an augmented knowledge sub-graph with different structural views. The objective is to identify items less sensitive to structure variations and more tolerant to connections with noisy entities and, as such, more helpful in capturing the preferences of correlated users. Furthermore, the component produces a knowledge graph structure consistency score c_i, which is defined as cosine similarity between embeddings created according to the relation-aware aggregation scheme defined above in Eq.3. According to [51], the lower c_i, the more item i is affected by the knowledge graph noise.

Knowledge-Guided Contrastive Learning uses the knowledge graph augmentations scheme and knowledge graph structure consistency score in the so-called interaction graph as well. The interaction graph is a graph of interactions between users and items. Similarly to KG, two structural views are constructed in order to find out which interactions to consider and which interactions to drop. The structure consistency score is used for defining a probability for masking vectors in the interaction graph correlated data views. The interaction subgraph augmented views are further corrupted by applying the aforementioned knowledge graph augmentation to consider relations between items, entities, and users. Afterwards, the user and item embeddings are achieved by utilizing a collaborative filtering framework as well as relation-aware knowledge aggregation. LightGCN [22] message propagation is employed to capture collaborative information (see also 2) while the aforementioned knowledge graph aggregation mechanism is used to enhance the item representation before it is fed to LightGCN component. Contrastive learning is then performed on the two created knowledge- and interaction-aware views with a loss function $\mathcal{L}_c$ based on InfoNCE [5]:

$$\mathcal{L}_c = \sum_{n \in \mathcal{V}} -\log \frac{\exp\left(s(\mathbf{x}_n^1, \mathbf{x}_n^2)/\tau\right)}{\sum_{n' \in \mathcal{V}, n' \neq n} \exp\left(s(\mathbf{x}_n^1, \mathbf{x}_{n'}^2)/\tau\right)} \tag{4}$$

where τ is the temperature parameter, $s(\cdot)$ is a similarity function between positive and negative pairs, and $\mathcal{V}$ are vertices in the graph.

Finally, the KGCL combines the contrastive learning loss function and BPR loss function [37] (see also Eq.1) to jointly learn the preference of users for items:

$$\mathcal{L} = \mathcal{L}_b + \lambda_1 \mathcal{L}_c + \lambda_2 \|\Theta\|_2^2, \tag{5}$$

where λ_1 and λ_2 are parameters that determine the strength of contrastive learning and regularization and Θ represents learnable model parameters.

CGCL. *Candidate-aware Graph Contrastive Learning* (CGCL) [16] enhances recommendation by constructing semantically meaningful contrastive pairs without relying on heuristic augmentations. It builds on LightGCN [17] and introduces three auxiliary losses: (1) a structural neighbour contrastive loss, which aligns a node with same-type neighbours across GCN layers; (2) a candidate contrastive loss, which pulls user embeddings closer to others who interacted with the same candidate item; and (3) a candidate structural neighbour contrastive loss, which models high-order collaborative signals by connecting users and candidate items through their respective neighbourhoods. Each loss follows an InfoNCE-style objective [34], and all are optimized jointly with a standard BPR loss [37] for ranking. This layered contrastive setup allows CGCL to improve representation learning, especially under data sparsity.

AdaGCL. *Adaptive Graph Contrastive Learning* (AdaGCL) [19] addresses structural noise in user-item interaction graphs by learning how to generate contrastive views dynamically. Instead of using static perturbations, AdaGCL employs two learnable modules: a variational graph autoencoder [20] that reconstructs the graph from latent variables to sample soft views, and a graph denoiser that filters edges based on learned scores using a concrete distribution [30]. These views are encoded with LightGCN and optimized via an InfoNCE contrastive loss applied to both users and items. The total objective combines this with BPR in a multi-task fashion. AdaGCL's adaptive augmentation improves robustness and performance on noisy or sparse datasets.

ALS based Matrix Factorization. Implicit feedback data causes complications when optimizing with standard gradient descent algorithms on squared error objective functions. [18] introduces a matrix factorization method where the optimization is based on an alternate least squares method with alternating computation of user and item factors with guarantees to lower cost function in each step. The method is further using confidence values for user preference.

UserKNN. UserKNN is a neighbourhood-based collaborative filtering method exploiting information from co-rated items or users who co-rated items. N most similar user or item rating vectors are considered to predict an item's recommendation/rating score. The neighbours are obtained by calculating a similarity between users (for example Pearson correlation) [33]:

$$sim(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in \mathcal{I}_{uv}} (r_{vi} - \bar{r}_v)^2}} \tag{6}$$

where $\mathcal{I}_{uv}$ are items commonly rated by user u and v, r_{ui} and r_{vi} are rating of user u and user v of item i respectively, $\bar{r}_u$ and $\bar{r}_v$ are user u and user v rating averages.

Once the neighbours are known, a prediction can be made as a weighted rating average of neighbouring user ratings as follows [33]:

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} sim(u,v) r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |sim(u,v)|} \tag{7}$$

where $sim(u,v)$ is a user similarity calculated by Eq. 6 and r_{vi} is user v rating of item i for which we predict a rating $\hat{r}_{ui}$ of user u.

It is a simple yet well-performing method in many cases. Since KNN is not an embedding-based method, we are not integrating it with review or cluster embedding.

Spectral Clustering Based Recommendation. Produces embedding as an eigenvector decomposition of the laplacian matrix L of the knowledge graph enhanced with user-item interaction (rating relation added) [45]:

$$L = D - A \tag{8}$$

where A is an adjacency matrix of the knowledge graph enhanced with user-item interactions, sometimes also called collaborative knowledge graph, and D is a diagonal matrix of node degrees.

By principal component analysis, we obtain eigenvectors and eigenvalues of the Laplacian matrix. We take k principal eigenvectors to form embeddings for each node of the knowledge graph. For recommendation, we only consider nodes of user and item types.

This is similar to how other matrix factorization methods work on rating data. The difference is that here, we consider rating interaction, knowledge graph relations, and eigenvector decomposition of a matrix constructed from the collaborative knowledge graph. In simple spectral clustering, we do not consider relation and node types. For recommendation purposes, we select embeddings of user and item nodes to perform DOT product or Euclidean Distance between them.

SpectralMix Based Recommendation. It is an unsupervised embedding approach that enhances spectral clustering by utilizing information from multiple types of edges, nodes, and node features. The goal of this method is to obtain an embedding of the nodes and node attributes in the graph so that similar objects (nodes and node attributes) are close and clearly separated from dissimilar ones. In this context, two nodes are similar if they are connected by many edge types. To utilize the node attribute information, they additionally consider a bipartite graph that links nodes with their corresponding categorical values of node attributes. This allows SpectralMix to obtain node and node attribute embeddings. The representation of nodes in SpectralMix is guided by its objective function [40]:

$$\mathcal{L} = \min_{O,\mathcal{M}} \sum_{r=1}^{|R|} \alpha_r \cdot \left(\sum_{e=(v_i,v_j,w_{ij},r)\in E_r} w_{ij} \cdot \sum_{l=1}^{d} \|o_{il} - o_{jl}\|^2 \right) + \sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{A}|} \alpha_j \cdot \sum_{l=1}^{d} \|o_{il} - m_{il}\|^2 \quad (9)$$
$$\text{subject to } O^T O = I_n.$$

where R represents the total number of graphs (edge types), O is a matrix of node embeddings, $\mathcal{M}$ matrix of node attribute embeddings, and $\mathcal{A}$ are attributes characterizing the nodes. Moreover, o_{il} and m_{il} represent the coordinates of the objects in the embedding space. α_r and α_j are weight coefficients for edge types and categorical node attributes. This objective function is based on Euclidean distance. In this study, instead of utilizing only the Euclidean distance, we experiment also with Cosine similarity (DOT product) and a combination of the Euclidean distance and DOT product. For recommendation purposes, we select user and item embeddings to perform prediction with DOT product or Euclidean distance. Euclidean distance is originally used as an objective function for learning in SpectralMix.

Review Based Recommendation. A heuristic method designed to investigate the impact of a single review-based embedding component in the recommendation. We construct an item-related review embedding as a mean aggregate of review text embeddings that are linked to that item. Similarly, to obtain a user-related embedding, we aggregate (mean) item-related review embeddings from all users related to the items the particular user has rated. Thus, we obtain a single embedding for each user and each item, respectively, as an input to calculate the prediction, as both DOT product and Euclidean distance between them. We utilize the SentenceTransformer model [36] for review text embeddings.

3.2 Model Learning and Prediction

Learning. In baseline methods, we follow learning as they are described in their respective papers and summarized above. We, where possible, use the code they provide with the methods.

For SpectralMix, which is designed for multi-relational graphs, we construct different relations between users and items. Also, we consider two versions of ratings to obtain embeddings. The first version learns only from movie ratings, and the second version learns from all ratings, including ratings on entities in the knowledge graph. In the other recommendation methods, we utilize their learning methods without modification, as mentioned by their authors. Further, we consider initializing the embeddings with a zero or a random vector.

Prediction. Each method, except KNN, produces embeddings of users and items. We test for two prediction methods: 1) **DOT product** of user and item embedding vectors as it is the standard method used for prediction in all state-of-the-art recommendation methods, and 2) **Euclidean distance** of user and item embedding vectors because it is used for the objective function in the SpectralMix. In the case of the KNN, we apply the standard weighted rating average of neighbours for prediction. The process is illustrated in Fig.1.

3.3 Integration of Cluster Centroid and Review Embeddings

Cluster Centroid Embeddings. Our aim is to investigate how helpful the cluster centroid embeddings are. We apply the K-Means algorithm to the embeddings obtained from each recommendation method, producing cluster centroids of similar users and items. We discuss in section 4 regarding the number of clusters parameter for the K-Means algorithm. We include two versions

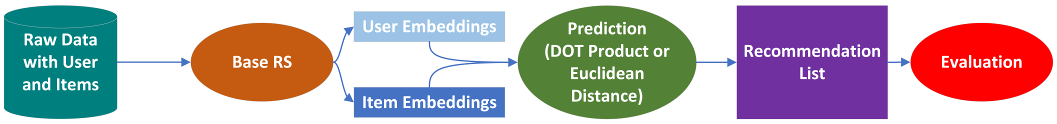
Raw Data with User and Items
Base RS
User Embeddings
Item Embeddings
Prediction (DOT Product or Euclidean Distance)
Recommendation List
Evaluation

Fig. 1. Prediction for base methods

of cluster centroid embeddings integration: 1) By concatenating user and item embeddings with the cluster centroid embeddings for a cluster where user and item belong to, respectively; 2) By computing DOT product between user and item embeddings and a cluster centroid embeddings of a cluster where user and item belong to, respectively. These enhanced user and item embeddings are used for prediction in the same way as described above (DOT product or Euclidean distance). The process is illustrated on Fig.2.

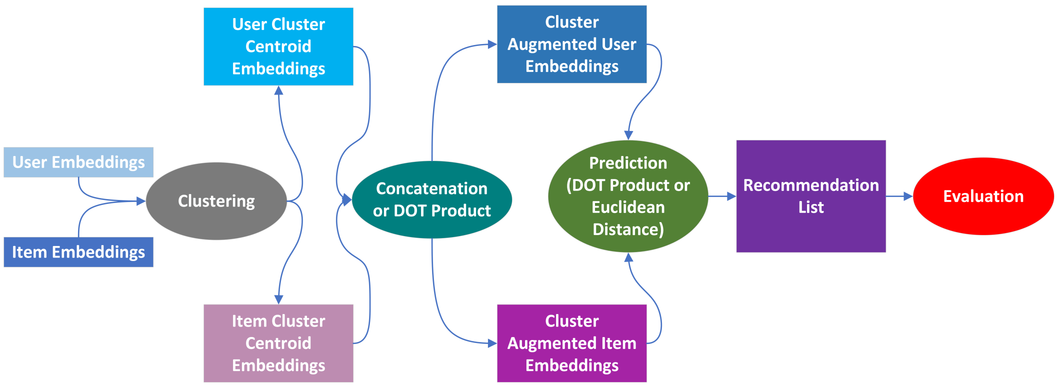
User Cluster Centroid Embeddings
User Embeddings
Clustering
Item Embeddings
Item Cluster Centroid Embeddings
Concatenation or DOT Product
Cluster Augmented User Embeddings
Cluster Augmented Item Embeddings
Prediction (DOT Product or Euclidean Distance)
Recommendation List
Evaluation

Fig. 2. Prediction for embeddings augmented with cluster centroid embeddings

Text Reviews Embeddings. We construct text (sentence) embeddings of the reviews using the widely used SentenceTransformer model [36][1]. The model is based on a BERT-like model [7] and is fine-tuned using contrastive learning on unsupervised data and, to construct hard negatives, natural language inference data.

The process of learning review embeddings and augmenting item and user embeddings with the review embeddings is illustrated in Fig. 3. Each item has up to N reviews. We compute an aggregated (mean) embedding of reviews for each item to integrate it into an item embedding. We concatenate such review embeddings to the corresponding item embeddings. Similarly, we aggregate (mean) review embeddings of items that a user has rated. Ideally, we would want to have reviews written by users from our repository, as in [8]. Unfortunately, this data is not available, and thus, we assume the average review for each user aggregated from item reviews such users gave a rating to. After the aggregation, we concatenate such aggregated review embeddings with user embeddings.

Integration. We also combine embeddings of cluster centroids, reviews, and recommendations. We consider two options: 1) We concatenate review embeddings with user and item embeddings enhanced with cluster centroid embeddings (cluster centroid embeddings are obtained before review embeddings are concatenated with user and item embeddings) - see Fig.4; 2) We cluster user

[1]Model available in the Transformers library[49] at https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

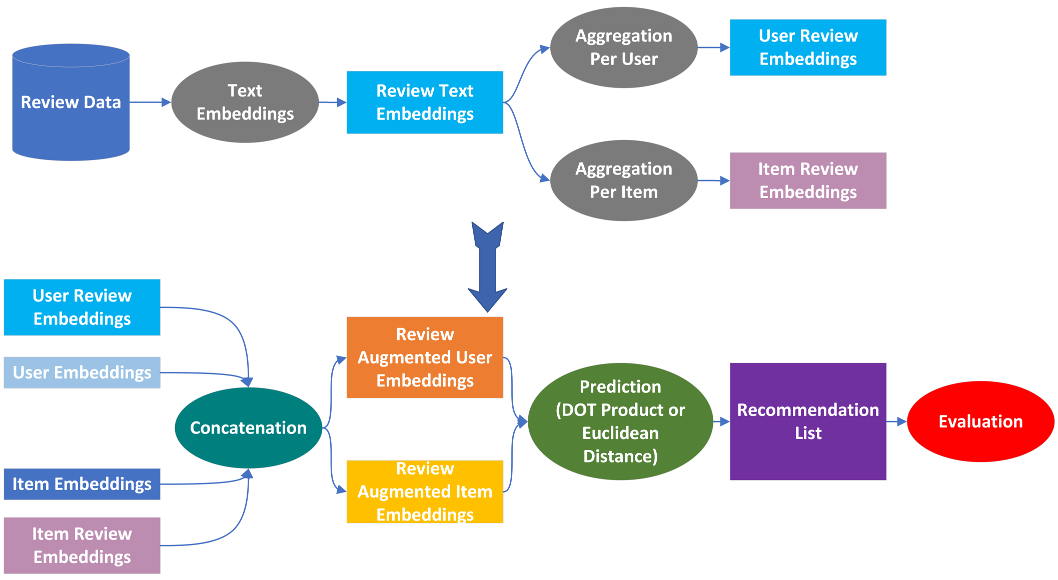

Fig. 3. Prediction for embeddings augmented with review text embeddings

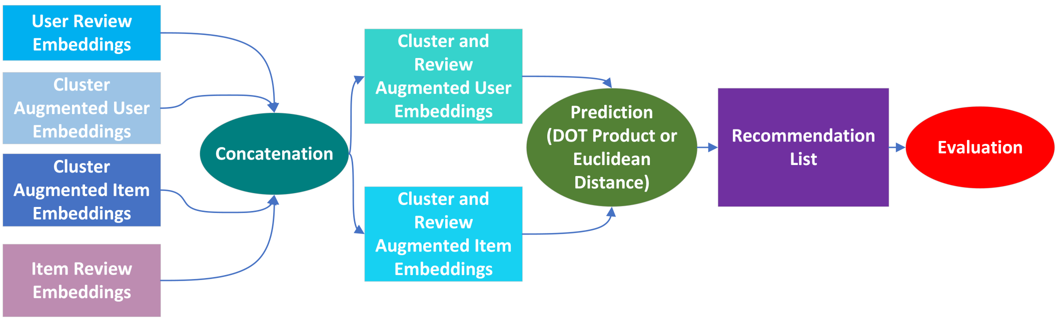
User Review Embeddings
Cluster Augmented User Embeddings
Cluster Augmented Item Embeddings
Item Review Embeddings
Concatenation
Cluster and Review Augmented User Embeddings
Cluster and Review Augmented Item Embeddings
Prediction (DOT Product or Euclidean Distance)
Recommendation List
Evaluation

Fig. 4. Prediction for embeddings augmented with cluster centroid and review embeddings

and item embeddings enhanced with review embedding (cluster centroid embeddings are obtained after the review embedding is concatenated with user and item embedding) - see Fig.5.

4 EXPERIMENTS

4.1 Datasets

We evaluate our solutions on the Mindreader dataset [3][2], Amazon Book dataset [15][3], and the Yelp dataset[4]. For comparison purposes and also due to the availability of the knowledge graph, we are using the version used also by KGCL [51][5]. Please note that we have also experimented with different versions of the Amazon Book and Yelp dataset by linking it to the knowledge graph from

[2]Available at: https://mindreader.tech/dataset/
[3]Available at: https://jmcauley.ucsd.edu/data/amazon/
[4]Available at: https://business.yelp.com/data/resources/open-dataset/
[5]Available at: https://github.com/yuh-yang/KGCL-SIGIR22/

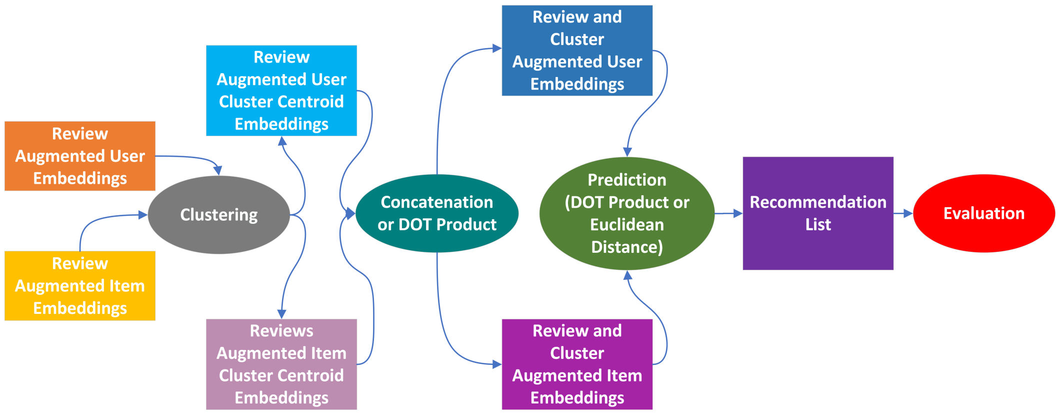
Review Augmented User Embeddings
Review Augmented Item Embeddings
Clustering
Review Augmented User Cluster Centroid Embeddings
Reviews Augmented Item Cluster Centroid Embeddings
Concatenation or DOT Product
Review and Cluster Augmented User Embeddings
Review and Cluster Augmented Item Embeddings
Prediction (DOT Product or Euclidean Distance)
Recommendation List
Evaluation

Fig. 5. Prediction for embeddings augmented with reviews and cluster centroid embeddings

the KGCL paper. Generally, this is possible, on the expense of lower coverage of entities from the knowledge graph, but the changes in results where negligible.

The Mindreader dataset. It features a knowledge graph in the movie domain and contains user ratings for movies and entities in the knowledge graph. It also contains explicit ratings for expressing an undecided preference and a negative rating. The version of the dataset we use contains 218.794 ratings from 2, 316 users, over 12, 206 entities, and an associated knowledge graph of 18, 133 movie-related entities. There are overall 120, 313 ratings of movies. of which there are 31, 734 positive rating. Please note that Mindreader has explicit ratings for positive, negative as well as undecided sentiment. The Mindreader positive rating density for movie ratings is 11.23×10^{-4} while the density of all explicit ratings on movies is 42.56×10^{-4}.

The knowledge graph is used for KGCL, Spectral Clustering, and SpectralMix. For KGCL, only positive ratings of movies are used since it cannot deal with other ratings. For SpectralMix, a full set of ratings is used to obtain embeddings, but only positive ratings for the prediction are considered, and 8 different relations between movies are constructed, such that two movies are connected if they share the same director, producer, genre, etc. . For LightGCN, CGCL, and AdaGCL, only the user-movie bipartite graph is used. For KNN and BPRMF, only the rating matrix derived from positive ratings of users for movies is used. We consider items and users that appear at least once in the testing fold for a particular fold of evaluation. In addition to the Mindreader dataset, we use a movie review dataset [29]. This dataset contains 50, 000 English movie reviews along with their associated sentiment labels "positive" and "negative". The link between movies and reviews is obtained through the same movie ID in both datasets.

The Amazon Book dataset. The dataset features user interactions and a knowledge graph from a book domain from the Amazon web shop. The version we used from [51] contains 70, 679 users, 27, 915 items and 846, 434 ratings of items with density of 4.8×10^{-4}. It contains an associated knowledge graph with 29, 714 entities and 39 relation types. The knowledge graph contains 686, 516 triples. The knowledge graph is used for KGCL and Spectral Clustering. For LightGCN, CGCL, and AdaGCL, only the user-movie bipartite graph is used. For KNN and BPRMF, only the rating matrix derived from positive ratings of users for books is used. In addition, we use book reviews from the dataset. This dataset contains 597, 232 book reviews along with their associated sentiment labels "positive" and "negative". The link between books and reviews is obtained using the same book ID.

The Yelp dataset. The dataset contains user interaction as well as user reviews of businesses. Also in this context, we use the version from [51], because it provides mapping to a knowledge graph of entities in the business domain. The dataset contains 45, 919 users and 45, 538 items and 1, 183, 610 ratings, and a density of 5.7×10^{-4}. Further, it provides a knowledge graph with 47 472 entities, 42 relations and 869, 603 triples. The knowledge graph is used for KGCL and Spectral Clustering. For LightGCN, CGCL, and AdaGCL, only the user-movie bipartite graph is used. For KNN, BPRMF, and ALS, only the rating matrix derived from positive ratings of users for books is used. In addition, we use business reviews from the dataset. The original review dataset contains 6, 990, 280 reviews. We only use those that match the item and user IDs available in the dataset form [51].

4.2 Number of Clusters

We used the Silhouette score [39] and Elbow analysis [42] to find an optimal number of clusters on the Mindreader dataset. The analysis suggested that the optimal number of clusters should be 3, 4, and 5. Additionally, we include $K = 2$ clusters to see how the performance changes, and we keep the same number of clusters for all considered recommendation methods. For uniformity, we use the same number of clusters also in experiments on the Amazon Book and Yelp datasets.

4.3 Metrics

We perform Top K recommendations evaluation on 5 folds for the Mindreader dataset and utilize LensKIT [10] for our evaluations. To split the data into folds we also use LensKIT random split function. In the case of Amazon Book and Yelp datasets, we perform experiments with the split provided by authors of the original KGCL paper [51]. We simply adopt their dataset version and their split as they have published it for comparison purposes. We however select all available items in the test sets in each dataset for constructing the recommendation list instead of random sampling.

We use standard Top K recommendation metrics provided by LensKIT. **Normalized Discounted Cumulative Gain (NDCG)@K** computes a mean utility score of ranked lists of recommendations of the size K. It values more if the relevant items are in the upper part of the list. **HIT@K** computes the fraction of the lists of the size K with at least one relevant item for a user appearing in a list generated for the user. **Precision@K**, denoted as P, computes a mean precision of ranked lists of a size K. **Recall@K**, denoted as R, computes a mean recall of ranked lists of size K. **Reciprocal Rank@K**, denoted as RR, computes the Reciprocal Rank of the first relevant item in the list of recommendations with size K. We evaluate our methods on ranked lists of 5, 10, 50, and 100 items.

4.4 Implementations

For baseline methods we have used their standard implementations in python as provided by the methods where possible and where not possible we have implemented them with the use of standard libraries[6] We have used their default parameter. We have imported the aforementioned dataset, run their learning, and extracted embeddings. We use the evaluation library provided by LensKIT after

[6]Libraries:
ALS: https://pypi.org/project/implicit/
KGCL: https://github.com/yuh-yang/KGCL-SIGIR22
LightGCN and BPRMF: https://github.com/gusye1234/LightGCN-PyTorch
Spectral Clustering: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.eigs.html
UserKNN: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html
SpectralMix: https://gitlab.cs.univie.ac.at/yllis19cs/spectralmixpublic
AdaGCL: https://github.com/HKUDS/AdaGCL
CGCL: https://github.com/WeiHeCnSH/CGCL-Pytorch-master
Review Embeddings/Transformer Model: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Table 1. Three best performing methods for Mindreader dataset (best one in bold)

Metric	K	ALGORITHMS			
		LightGCN-DP-R	LightGCN-DP	KGCL-DP-R	UserKNN
NDCG	5	**0.22173**	0.17957	0.13507	-
	10	**0.25421**	0.21949	0.15129	-
	50	**0.32348**	0.29490	0.19591	-
	100	**0.35000**	0.31923	0.22537	-
HIT	5	**0.06143**	0.04968	0.04095	-
	10	**0.07836**	0.07084	0.05227	-
	50	0.12225	0.12009	-	**0.17511**
	100	**0.12009**	0.13521	-	**0.32386**
P	5	**0.01405**	0.01197	0.00857	-
	10	**0.00949**	0.00919	0.00555	-
	50	0.00371	**0.00378**	-	0.00374
	100	0.00234	0.00233	-	**0.00373**
R	5	**0.26115**	0.22956	0.15455	-
	10	**0.35323**	0.34174	0.20042	-
	50	0.65873	**0.66371**	0.39792	-
	100	**0.81225**	0.80535	0.57278	-
RR	5	**0.04040**	0.02626	0.02726	-
	10	**0.04265**	0.02909	0.02874	-
	50	**0.04468**	0.03149	0.03041	-
	100	**0.04490**	0.03171	0.03076	-

we have performed predictions on all described combinations with reviews and cluster centroids. We have implemented prediction functions and embedding combination methods ourselves. We also provides various utilities to simplify the process of evaluation and preprocessing[7].

4.5 Results

To illustrate the performance of the methods we choose to present the first 3 best methods in each measure. Then we also utilize bar graphs where we select the top 15 methods in a selected metric and plot their performance in each considered metric. For the names of methods which follow the application of steps in our methodology, we use the following abbreviations:

(1) DP – DOT Product;
(2) ED – Euclidean Distance;
(3) KC – K is a number and C means clusters, meaning the number of clusters;
(4) CC – Concatenating cluster centroids;
(5) DC – DOT product with cluster centroids;
(6) R – Concatenating Reviews;
(7) OF-Method – Objective function in SpectralMix with DP, ED, or DP x ED.

Under the Mindreader dataset, further to the above, for the SpectralMix method in learning we have considered 2 seeds ($S0$ — initializing with zero vector or $S1$ — initializing randomly) and learning from all ratings (*AR*) including ratings on entities, or only movie ratings (*MR*).

Top Performers

Mindreader dataset. Table 1 presents the best 3 performing methods under each k for Mindreader dataset. We can see that the best performing method across all the K values under NDCG is LightGCN-DP-R, LightGCN augmented with review embeddings, followed by Light-GCN in a

K-Means Clustering: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
Evaluation from LensKIT: https://lenskit.org/
[7]Our code: https://github.com/peterdolog/RecommendationsWithClusteringAndReviews

Table 2. Three best performing methods for Amazon Book dataset (best one in bold)

Metric		ALGORITHMS								
	K	KGCL-DP CC-R-4C	Spect.Cl. DP-R	KGCL-DP CC-R-3C	AdaGCL-DP CC-R-2C	LightGCN DP-CC-R-5C	KGCL DP-R	KGCL DP	AdaGCL DP-R	LightGCN DP-CC-R-4C
NDCG	5	**0.04944**	0.04943	0.04943	-	-	-	-	-	-
	10	0.05824	-	-	**0.05826**	0.05823	-	-	-	-
	50	-	-	-	0.07657	-	**0.07847**	0.07786	-	-
	100	-	-	-	-	-	**0.09317**	0.09258	0.09103	–
HIT	5	0.09673	-	0.09673	-	-	-	-	-	**0.09676**
	10	-	-	-	0.13714	-	**0.14375**	0.14330	-	-
	50	-	-	-	-	-	**0.37107**	0.36925	0.35806	-
	100	-	-	-	-	-	**0.47941**	0.47864	0.47791	-
P	5	**0.02090**	-	0.02090	-	-	-	-	-	0.02090
	10	-	-	-	0.01515	-	**0.01584**	0.01577	-	-
	50	-	-	-	-	-	**0.00971**	0.00967	0.00921	-
	100	-	-	-	-	-	**0.00697**	0.00696	0.00689	-
R	5	**0.06092**	-	0.06091	0.06090	-	-	-	-	-
	10	0.08602	-	-	**0.08609**	-	-	-	-	0.08599
	50	-	-	-	-	-	**0.22449**	0.22333	0.20989	-
	100	-	-	-	-	-	**0.30696**	0.30584	0.30138	-
RR	5	-	**0.05568**	0.05563	-	-	-	-	-	0.05564
	10	-	**0.06099**	-	0.06096	-	-	-	-	0.06096
	50	-	0.06688	-	**0.06691**	-	-	-	-	0.06687
	100	-	0.06798	-	**0.06801**	-	-	-	-	0.06798

standard form (with DOT product as a prediction function) and followed by KGCL-DP-R, KGCL augmented by review embeddings. We can see that in top places there is no method with clustering and in addition, the second best method is an ordinary collaborative filtering graph convolutional network LightGCN. This is because the Mindreader dataset has a reasonably large number of ratings. It is still sparse, but the sparsity is lower than for example the Amazon Book dataset.

Similar order in performance can be seen in other measures. Interesting exceptions are $HIT@50$, $HIT@100$, and $P@100$ where UserKNN scores the highest. Again, this is a collaborative filtering method, benefiting from a sufficient number of ratings in the dataset. The reason for good performance is likely a good ability to group user neighbours and get a sufficiently good recommendation for many users (HIT measure) as well as making it to a sufficient number of lists of top 100 items for each user.

Amazon Book dataset. Table 2 presents the best 3 performing methods under each K for the Amazon Book dataset. We can see that, under NDCG measure, the best performing method for $K = 5$ is KGCL-DP-CC-R-4C, KGCL augmented with concatenated cluster centroids with 4 clusters and review embeddings, followed by Spectral Clustering based method augmented with reviews and followed by KGCL augmented with concatenated cluster centroids with 3 clusters and review embeddings (KGCL-DP-CC-R-3C). Having the Spectral Clustering method, though augmented with review embeddings so high is surprising but shows that review embeddings provide a good signal even for simpler methods. In the case of $K = 10$ we have AdaGCL-DP-CC-R-2C on the top place, AdaGCL augmented with concatenated clusters, with 2 clusters, and review embeddings, followed by KGCL-DP-CC-R-4C and LightGCN-DP-CC-R-5C which is augmented LightGCN method. At the $K = 50$ and 100 the top performer is KGCL augmented with reviews, followed by a baseline KGCL. The 3rd position for those cases is taken by AdaGCL, augmented with cluster centroids and reviews for $K = 50$ and with review embeddings only for $K = 100$. We can see that KGCL is the strongest method here. The knowledge graph provided for the Amazon book dataset plays a role here. We can see it also in the case of Spectral Clustering for NDCG@5. We can also see that for smaller lists, cluster centroids play a larger role while for larger lists especially with KGCL even a baseline method is sufficient, and for methods not using knowledge graphs, augmentation with review embeddings is sufficient.

It is also worth noticing, that differences in performance of top performing methods are rather small which in a practical sense may mean that it is probably not so important which of those methods we are choosing on the more sparse datasets, but it is rather the importance of clustering and review embeddings together that makes a difference. In general, KGCL derived method and even sometimes KGCL baseline perform the best in other measures as well.

The only exception for the best performing method is the $HIT@5$ where LightGCN-DP-CC-R-4C performs the best. Again here we see that it is a combination of clustering, and review embeddings but on collaborative information without a knowledge graph. In addition, for $HIT@10$, $HIT@50$, and $HIT@100$, we can find AdaGCL derived methods in the third place. As for the Reciprocal Rank measure, as can be seen in Table 2, SpectralClustering-DP-R, spectral clustering-based method augmented with review embedding, performs the best followed by LightGCN derived method and then KGCL derived method. Here we see a simpler method again performing well, but again a method that uses a knowledge graph of side information. On the Amazon dataset, due to the larger sparsity of collaborative information, in general, any additional information helps to improve the recommendation. But it depends on the measure which method to choose. Please also note that we were not able to run the SpectralMix method on the Amazon Book dataset due to memory issues.

Yelp Dataset. As can be seen for Table 3, the situation in the Yelp dataset is different from the Amazon Book and the Mindreader datasets. The top performers on all measures are either AdaGCL or KGCL derived methods. AdaGCL derived methods perform the best on smaller lists ($K = 5$ and 10) while KGCL derived methods perform the best on larger lists $K = 50$ and 100 and they simply exchange their positions largely due to the differences in their contrastive learning approach. It is also interesting to see that at the top performance places are baseline methods or methods augmented with review embeddings. We can also see that cluster centroids do not play a large role here. Moreover, we can see that review embeddings are of larger importance in the case of smaller lists while in the case of larger lists, it is purely the KGCL contrastive learning on knowledge graph together with collaborative information from its LightGCN backbone. Similarly to the Amazon Book dataset, we can see that the differences in performance are rather small and therefore, it may be sufficient to use simpler methods augmented with clustering and review embeddings. Similarly here, we were not able to run the SpectralMix due to memory issues.

Performance Beyond Top 3 Methods

Mindreader Dataset. Fig. 6 shows experimental results for selected methods for varying sizes of recommendation lists ($K = 5$, 10, 50, and 100) on Mindreader. We have studied all mentioned combinations of methods. For the effective presentation, we have selected the top 15 methods sorted according to NDCG@5. Here we can see a similar trend as in the top 3 recommendations. On top places are various derivations of LightGCN. Variants include also clustering augmented embeddings. The collaborative information and aggregation of such information from neighbours in convolutions of LightGCN serve as sufficient evidence for good recommendations. SpectralMix derived methods are also included in the top 15 methods ordered according to $NDCG@5$. This shows, that knowledge graph structure is important and can have benefits on the recommendation methods. Further, SpectralMix as a more complex method than Spectral Clustering performs better here. We can see that in general, for SpectralMix, seed 1 performs better, and learning from all ratings performs better. Please also note a few differences in comparison to the presentation we have chosen in [47]. In [47], in the candidate set for the top 10 performing methods we have

Table 3. Three best performing methods for Yelp dataset (best one in bold)

Metric	K	ALGORITHMS			
		AdaGCL-DP-R	AdaGCL-DP	KGCL-DP-R	KGCL-DP
NDCG	5	**0.02272**	0.02226	0.02144	-
	10	**0.02848**	0.02788	0.02714	-
	50	0.05104	-	**0.05241**	0.05201
	100	0.06635	-	**0.06857**	0.06805
HIT	5	**0.08835**	0.08691	0.08515	-
	10	**0.15331**	0.15048	0.14915	-
	50	0.39639	-	**0.41615**	0.41371
	100	0.53928	-	**0.56243**	0.55918
P	5	**0.01873**	0.01839	0.01797	-
	10	**0.01724**	0.01692	0.01664	-
	50	0.01149	-	**0.01218**	0.01210
	100	0.00933	-	0**0.00986**	0.00979
R	5	**0.02477**	0.02432	0.02367	-
	10	**0.03914**	0.03825	0.03783	-
	50	0.11959	-	**0.12778**	0.12678
	100	0.18980	-	**0.20206**	0.20041
RR	5	**0.04358**	0.04296	0.04182	-
	10	**0.05208**	0.05130	0.05016	-
	50	**0.06291**	0.06201	0.06211	-
	100	**0.06495**	0.06403	0.06419	-

chosen the best performing methods from all configurations and in addition the ones closest to the configuration which performs overall. In this paper, we have simply just ordered the methods according to $NDCG@5$ and then selected the top 15 most performing methods. We could do it this way here because we have more space to present also other views and tables while in [47] we had severe space limitation and we wanted to have coverage over all aspects of the performance in one bar chart. Therefore, for example, we do not have UserKNN represented in Fig. 6, since it performs the best in $HIT@K$ measure but not in $NDCG@5$ measure. We still needed to choose a subset of methods, because there are altogether 615 combinations of methods we have studied, which is not possible to present in one bar chart[8].

In Table 1, we report that it is UserKNN that performs the best. We have therefore produced a bar graph where we selected the top 15 methods sorted according to $HIT@50$ and show only performance for HIT measure. The bar chart is presented in Fig. 7. We can see here, that UserKNN performs in general relatively well on this measure, but the best on the $HIT@50$ and $HIT@100$. Here all top 3 methods are collaborative. The rest of the methods in the top 15 list are derivations of LightGCN, KGCL, and SpectralMix.

Please also note, that more complex contrastive learning approaches such as CGCL or AdaGCL are not present in the top performing methods. This is likely because the Mindreader dataset is rather small, and does not have enough samples to support the kind of learning those methods propose.

Amazon Dataset. Similarly to the Mindreader dataset, we wanted to see performance beyond the top 3. Figure 8 shows experimental results for selected methods for varying sizes of recommendation lists (k = 5, 10, 50, and 100) for the Amazon Book dataset. Also here, we have selected the top 15 methods sorted according to NDCG@5. Beyond the top 3 methods, which are all based on knowledge graphs with entities we also see LightGCN but this time both, clustering and review

[8]You can find a CSV file with results for all tested combinations for the Mindreader dataset at https://github.com/peterdolog/RecommendationsWithClusteringAndReviews/blob/3ea2370311b6cda1a430a406bfa28b011a78471a/EvaluationResults/AggregatedAll_Mindreader_ndcg_5.csv

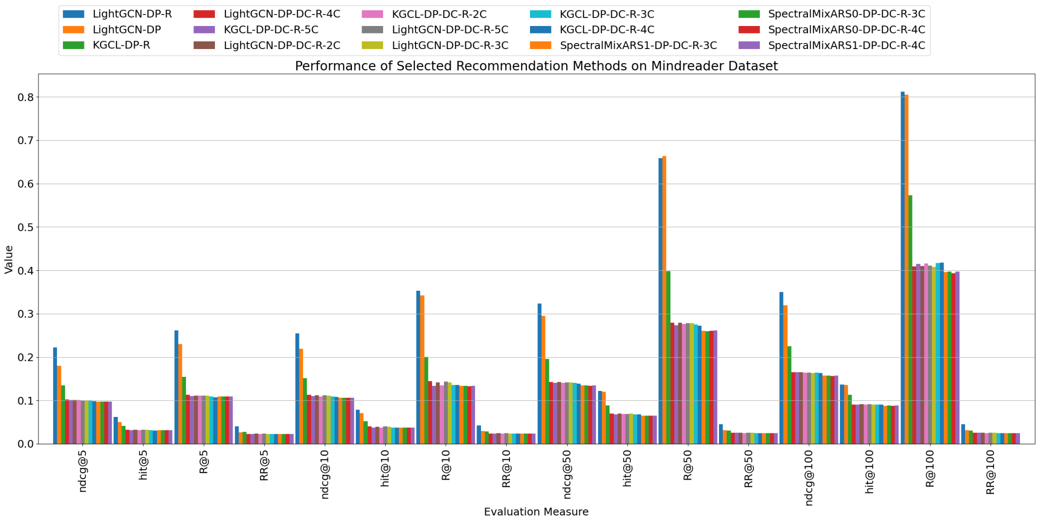

Fig. 6. Performance for the selected methods and their integration with clustering, review embedding, and their combination on the Mindreader dataset. Top 15 selected according to NDCG@5.

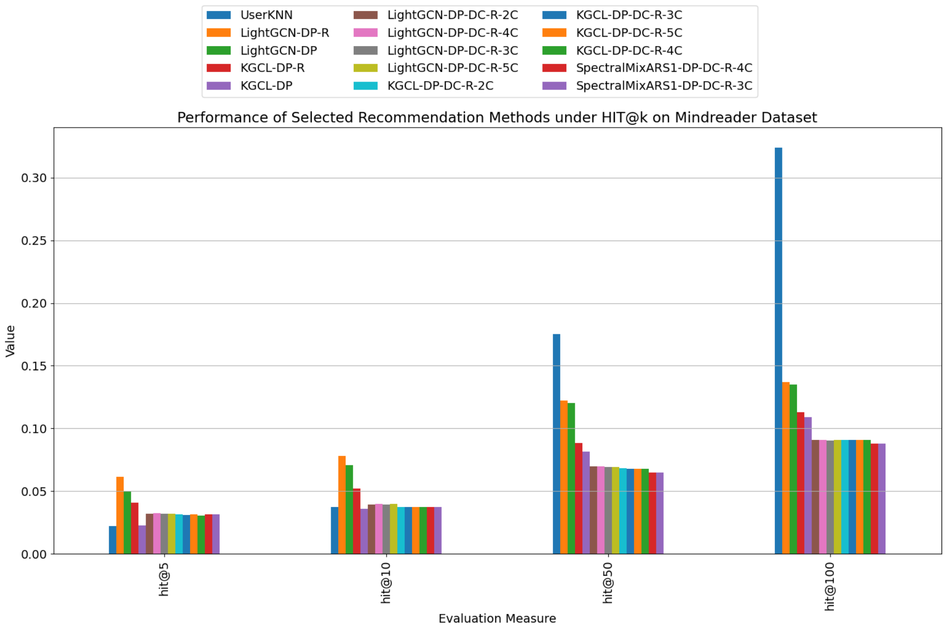

Fig. 7. Performance for the selected and their integration with clustering, review embedding, and their combination on the Mindreader dataset. Top 15 selected according to HIT@50.

embeddings needed to be used to score higher. We can also see that among the top 15 we also find the BPRMF method augmented with clustering and review embeddings. We can also see the other two contrastive learning methods, CGCL and AdaGCL there, both requiring augmentation

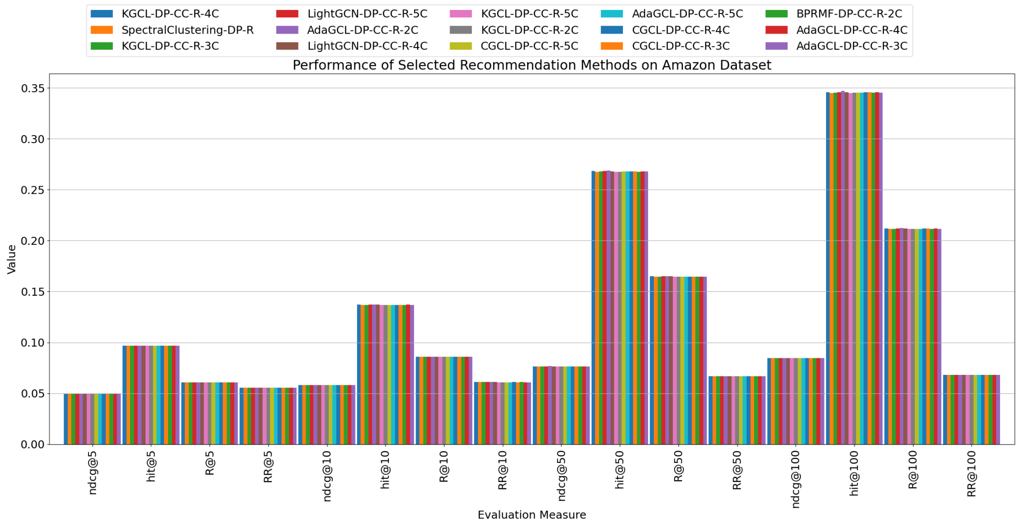

Fig. 8. Performance for the selected and their integration with clustering, review embedding, and their combination on the Amazon Books dataset. Top 15 selected according to NDCG@5.

with reviews and cluster centroids. The conclusion here shows that it is no longer sufficient to have a collaborative signal. For collaborative methods, both, clustering and review information is needed to bring them up in performance. For methods that already consider side information

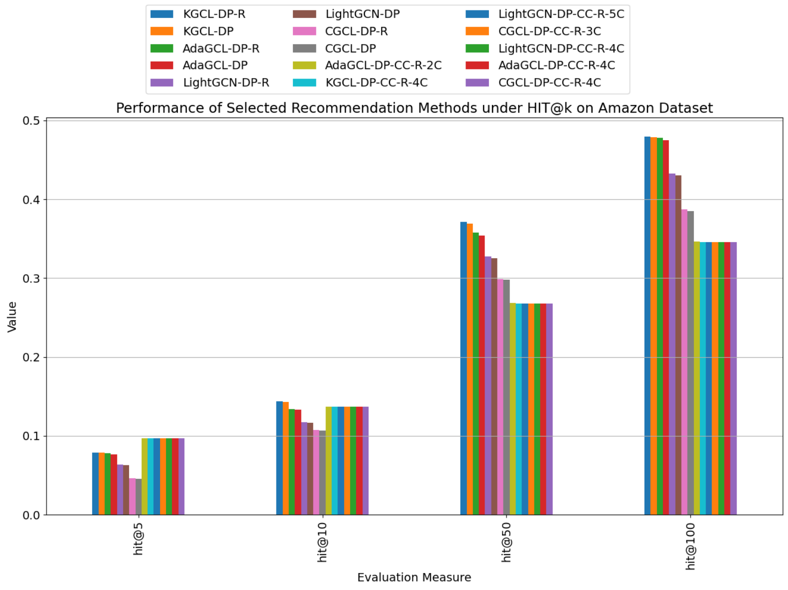

Fig. 9. Performance for the selected and their integration with clustering, review embedding, and their combination on the Amazon Books dataset. Top 15 selected according to HIT@50.

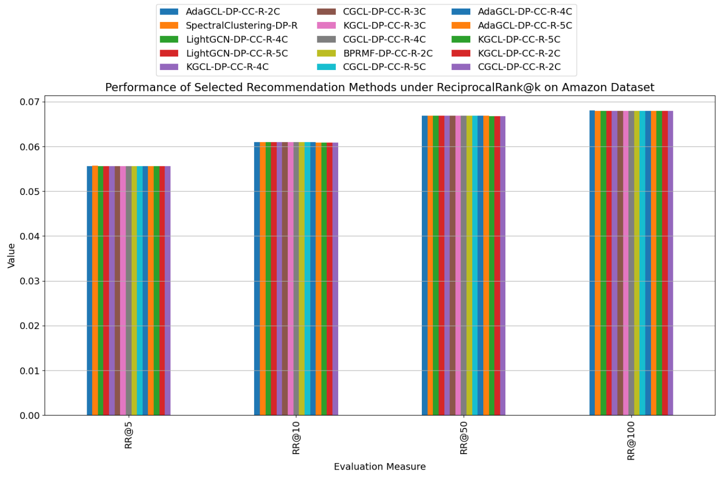

Fig. 10. Performance for the selected and their integration with clustering, review embedding, and their combination on the Amazon Books dataset. Top 15 selected according to RR@50.

in the form of a knowledge graph, it is sometimes sufficient to use reviews such as in the case of SpectralClustering, or simply use a baseline as in the case of KGCL baseline which still appears in the top 15. But top performing method on this chart is still a method that combines all the information we have considered. Again, it is not possible to present all results here effectively[9].

We can see already in Fig.8 but also in Table 2, that there are some differences in top performing methods under HIT@K in comparison to NDCG@5. Therefore and also to keep uniformity we have also drawn a bar chart for the top 15 performing methods sorted according to HIT@50. Fig. 9 shows the methods under the HIT@K measure for varying K. Here we can see that while under HIT@5 it is the LightGCN augmented with clustering and reviews with 4 clusters that perform the best, it is the KGCL augmented only with reviews that perform the best from $K = 10$ and upwards.

We could see in Table 2 that SpectralClustering augmented with review embeddings performs the best under $K = 5$ and 10 for the Reciprocal Rank measure. To see how it looks beyond the top 3 performing methods we have also drawn a bar chart for the Reciprocal Rank measure which is displayed in Fig. 10. Here we can see, similarly to other figures that it is a mix of methods. Among the top performing ones are also LightGCN, KGCL, AdaGCL, and CGCL. KGCL, and AdaGCL perform well in the base form, and when augmented with cluster centroids and review embeddings. We can also spot BPRMF but augmented with cluster centroids and review embeddings.

Yelp Dataset. Yelp dataset is showing a slightly different picture. Fig. 11 shows that ALS based method is the only one among top 15 methods besides the ones derived from KGCL or AdaGCL on the Yelp dataset. Also, it is clear from the figure that there is a large difference in performance after the top 4 methods. This says that the role of contrastive learning on the knowledge graph and the collaborative graph is playing the most important role in the Yelp dataset. Review embeddings are also helping methods here too. Interestingly, AdaGCL and KGCL derived methods with Euclidean Distance as a prediction function perform among the top 15 tested recommendation methods.

[9]You can find a CSV file with results for all tested combinations for the Amazon dataset at https://github.com/peterdolog/RecommendationsWithClusteringAndReviews/blob/3ea2370311b6cda1a430a406bfa28b011a78471a/EvaluationResults/AggregatedAll_Amazon_ndcg_5.csv

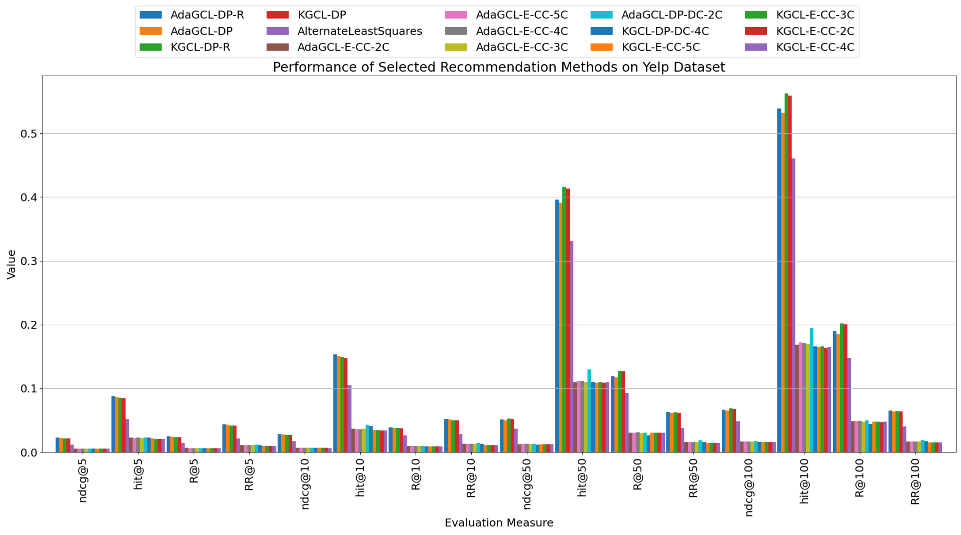

Fig. 11. Performance for the selected and their integration with clustering, review embedding, and their combination on the Yelp dataset. Top 15 selected according to NDCG@5.

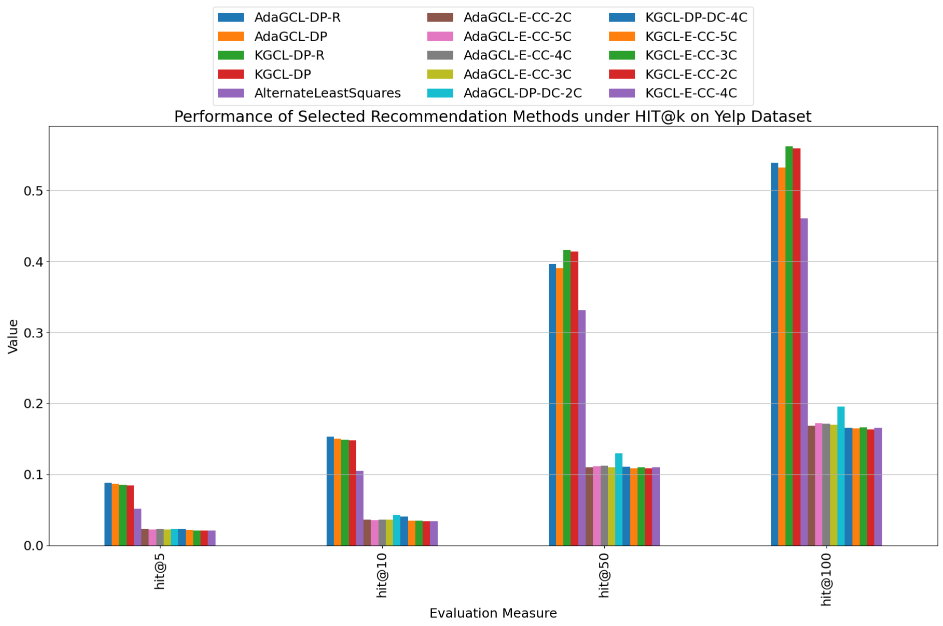

Fig. 12. Performance for the selected and their integration with clustering, review embedding, and their combination on the Yelp dataset. Top 15 selected according to HIT@50.

Fig. 12 shows the top 15 methods under HIT@K measure. Similarly to the Mindreader dataset, we can see here that UserKNN method performs relatively well (among the top 15). This can be explained by the fact, that the version of the Yelp dataset we used has a bit larger density than the

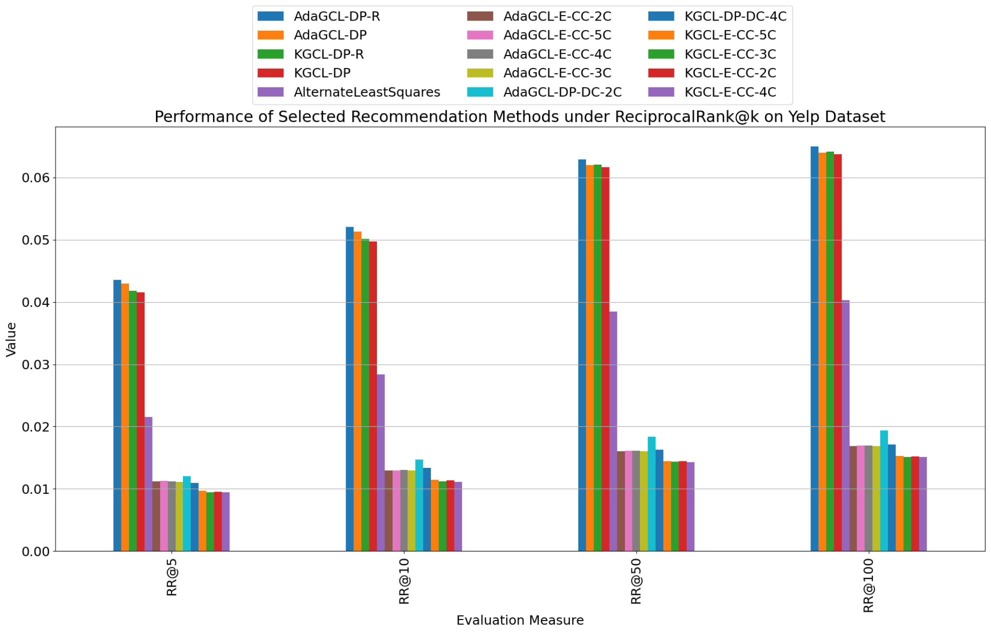

Fig. 13. Performance for the selected and their integration with clustering, review embedding, and their combination on the Yelp dataset. Top 15 selected according to RR@50.

Amazon Book dataset. On the other hand, it has a smaller density than the Mindreader dataset, and therefore UserKNN is not the best method under some K but only one of the top 15 methods. But this is still a rather surprising indication that even simple methods are still good and sometimes can be practically good enough for making recommendations especially when we have rather active users in the system. Fig. 13 shows a similar picture. But again, the differences in performance of methods after the top 4-5 are rather large. Further results can be found in the complete result set [10].

Baseline Methods

By baseline methods here we mean original methods without augmentation with reviews and cluster embeddings such as LightGCN, BPRMF, KGCL, AdaGCL, CGCL and so on.

Table 4 shows the performance of the baseline methods on the Mindreader dataset. We can see that the LightGCN performs the best followed by KGCL and ReviewOnly based heuristics. This also confirms that on the denser dataset, the collaborative information from ratings and/or from reviews is influential, and additional side information and graph structure on that side information does not help as much. We can of course conclude, that graph structure on ratings relation is of course important since the convolutions in LightGCN on the collaborative bipartite graphs give the method an edge. Interestingly, the simple ReviewOnly-based heuristic performs very well only a little worse than the KGCL method.

Table 5 presents the performance of the baseline methods on the Amazon Book dataset. For this table, we have selected NDCG@10 and Recall@10 measures. In general, we can observe that

[10]You can find a CSV file with results for all tested combinations for Yelp dataset at https://github.com/peterdolog/RecommendationsWithClusteringAndReviews/blob/3ea2370311b6cda1a430a406bfa28b011a78471a/EvaluationResults/AggregatedAll_Yelp_ndcg_5.csv

Table 4. Performance of baseline methods under NDCG@K and R@10 for Mindreader dataset

ALGORITHM	NDCG@10	R@10
LightGCN-DP	0.21949	0.34174
KGCL-DP	0.08919	0.14511
ReviewOnlyAR-DP	0.08730	0.09918
SpectralMixARS1-DP	0.03277	0.05722
SpectralMixARS0-DP	0.03260	0.05701
SpectralMixMRS1-DP	0.02684	0.05081
SpectralMixMRS0-DP	0.02684	0.05081
AdaGCL-DP	0.02089	0.03672
SpectralMixOPDPxEDARS0-DP	0.01534	0.02725
SpectralMixOFDPARS0-DP	0.01518	0.02805
BPRMF-DP	0.01335	0.02292
SpectralMixOFDPARS1-DP	0.01244	0.02277
SpectralMixOPDPxEDARS1-DP	0.01209	0.02159
UserKNN	0.01074	0.01951
SpectralClustering-DP	0.00900	0.01647
CGCL-DP	0.00870	0.01703
AlternateLeastSquares	0.00109	0.00310

Table 5. Performance of baseline methods under NDCG@K and R@10 for Amazon Book dataset

ALGORITHM	NDCG@10	R@10
KGCL-DP	0.04440	0.08103
AdaGCL-DP	0.04177	0.07182
LightGCN-DP	0.03541	0.06304
AlternateLeastSquares	0.03065	0.05535
CGCL-DP	0.02887	0.06284
BPRMF-DP	0.01990	0.03955
SpectralClustering-DP	0.00174	0.00188
UserKNN	0.00057	0.00072
ReviewsOnly-DP	0.00047	0.00080

we achieved a bit lower performance for KGCL while for the LightGCN baseline, it is a bit higher than reported in the literature or when running it with the evaluation directly in their code. The difference in performance in comparison to the numbers reported in both mentioned papers is likely caused by different ways of selecting candidates for recommendation lists in the evaluation. We are selecting all books in the test set for every user when constructing a recommendation list. Nevertheless, we confirm the order of the method's performance – at least between KGCL and LightGCN. Here we can see, that due to higher sparsity, the graph structure of side information has a better value since KGCL is outperforming other methods. On the contrary to the Mindreader dataset, the simple ReviewOnly-based heuristic is not performing as well as under the Mindreader dataset. We can speculate that the coverage of the reviews is not as high as in the Mindreader dataset. As mentioned above, we could not get the SpectralMix into the table due to performance problems on the Amazon Book dataset. As for AdaGCL, the paper does not reveal the results on the Amazon Book dataset. We were not able to fully reconstruct the results of CGCL on the Amazon Book dataset. The performance we have achieved is lower. In [16] we can see that the version of the Amazon Book dataset they use has a larger density which means a larger number of user-item interactions. Also, the evaluation methodology is different from ours since they use a temporal splitting. All this means that in our case the reported number is smaller. We have seen the influence of the density/sparsity on the performance in this paper since the performance differs on different datasets selected here. In addition to that, we can also see now that evaluation methodology and how the data is split into testing and training set has an influence on the performance of the methods.

Table 6. Performance of baseline methods under NDCG@K and R@10 for Yelp dataset

ALGORITHM	NDCG@10	R@10
AdaGCL-DP	0.02788	0.03825
KGCL-DP	0.02690	0.03739
AlternateLeastSquares	0.01702	0.02625
UserKNN	0.00859	0.01727
CGCL-DP	0.00193	0.00261
LightGCN-DP	0.00046	0.00064
BPRMF-DP	0.00043	0.00059
SpectralClustering-DP	0.00012	0.00018
ReviewsOnly-DP	0.00011	0.00016

Table 6 presents the performance of the baseline methods on the Yelp dataset. Here, the best performing method is AdaGCL. KGCL performs second best. We could see that the results we have achieved are comparable to those reported in the AdaGCL paper. In comparison to KGCL paper, our results are slightly lower, likely for the same reason as in the case of the Amazon Book dataset, i.e. difference in how we select the candidate set. We could not reconstruct the results for the CGCL similarly to Amazon for the same reasons. Please note, when running it inside their own method, the results are as they report since they utilise their dataset temporal split and their temporal-based negative sampling. Here we note, similarly to Amazon Dataset, that both, the sparsity as well as testing methodology has an influence on the results.

Discussion of Impact of Various Components

Impact of Collaborative Information. The collaborative information has by far the largest influence when the dataset is less sparse as shown by the best performing method, LightGCN augmented with reviews on Mindreader, but also by the performance of AdaGCL and UserKNN on the Yelp dataset. The LightGCN considers a bipartite graph of user ratings on movies. Even the original prediction method without integration with review embeddings achieves the second-best performance, except when overperformed by the KGCL on the Reciprocal Rank metric on larger lists and the UserKNN on the HIT metric on larger lists. The high dimensional rating vectors and the prediction of the weighted rating average are capable of satisfying more users with at least one relevant recommendation.

The collaborative information remains impact-full even in less sparse datasets, but there, more side information is needed. On the other hand, AdaGCL contrastive learning mechanism is helping to squeeze as much as possible from the collaborative information and review embeddings is sufficient in datasets such as Yelp.

As can be seen on the Amazon Book dataset, the best performing method under $NDCG@5$ was a derivative of KGCL with all considered components. Even on other measures, it was at least a graph with entities, collaborative information, and reviews which were needed to perform best.

Impact of Graph Structure. As already pointed out, we can see that methods that utilize graph structure with side information such as entities in knowledge graphs are beneficial for more sparse datasets such as Amazon Book. But graph structure seems important even in less sparse datasets such as Mindreader and Yelp where it is used on collaborative information. Methods such as LightGCN and AdaGCL seem to perform well. CGCL on the other hand can perform well when time stamp information is available.

Impact of Review Embeddings. Adding the information provided by review embeddings improved all the methods, and the improvement is rather large. On the Mindreader dataset, we can also note that the 3 best performing results include methods augmented only with review embeddings. That alone says that reviews are a valuable source of information and often even better than some established methods, such as UserKNN or some combinations of SpectralMix with clustering and review embedding. This is natural since the reviews contain rich information about user preferences, and when connected with ratings, it can further amplify the user preference insights.

On the Amazon dataset, the impact of the review embeddings component is also large. But to perform the best, it was not sufficient alone and needed to be enhanced with either graph structure of entities, or in addition to that with clustering.

On the Yelp dataset, the review component was also contributing with positive performance, but since the dataset was larger in comparison to Mindreader, and denser in comparison to Amazon Book, even contrastive learning methods in base form performed well enough.

Impact of Cluster Centroid Embeddings. On the Mindreader dataset, in the LightGCN, the clustering applied on embedding from collaborative rating makes the performance worse. KGCL performance is improved by integrating the text review embeddings, but the integration of the clustering makes it worse. It seems that clustering on embedding from collaborative information can not help to catch hidden information but rather introduce contradictory information into the recommendation process. Methods based on graph cuts or non-neural methods, such as BPRMF, benefit from text review embeddings. Additionally, clustering on integrated information from collaborative, knowledge-based, and review sources provides the best performance. Thus, clustering helps to reduce the noise in those methods.

We can see a much larger impact of clustering on the Amazon Book dataset but it helps the most when it is combined with review embeddings. The clustering seems to reduce the noise and missing information from collaborative information on the Amazon Book dataset and when extended with reviews it improves overall performance. On the Yelp dataset, the impact of clustering is not as large and typically helps in methods which are not performing as well in the base form.

Performance Considerations. Time complexity of the baseline methods are typically discussed in the source papers. In principle, each additional component to the collaborative backbone adds to the time complexity. For example, CGCL method considers 5 different components in objective functions. Similarly, KGCL considers several ones. Adding additional review embeddings and cluster centroid embeddings also raises the complexity of the methods. Review embeddings have the typical complexity of transformers while clustering complexity has the typical complexity of clustering algorithms which is dependent on the number of clusters and number of data points as well as the dimensionality of data points. We can also see, that many times, the clustering and review embeddings are more helpful in simpler methods which do not consider additional terms or objective functions in joint learning. This means that they can be used with advantage in those methods and overall still the composite method will have lower time complexity than more complex methods. Therefore, depending on the dataset, sparsity and coverage of the reviews, practitioners should see where and when they could use such additions. Adding them to the existing method is not complicated and they are well understood.

5 CONCLUSIONS AND FUTURE WORK

We provided a study on the impact of graph structure, clustering and review embedding on the performance of selected state-of-the-art recommendation methods. We have shown that graph structure, clustering and review embeddings positively impact performance. There are some differences in how in practice the methods should be selected since the performance slightly differed between datasets. It seems that graph structure on side information in knowledge graph is more important on sparser datasets while it is sufficient to consider graph structure on collaborative information in denser datasets. It also seems that the differences in performance on sparser datasets

Table 7. Key Findings on the Impact of Clustering, Graph Structure, and Review Embeddings

Dataset	Most Interesting Observations
Mindreader (denser)	• Best performance from LightGCN-DP-R (collaborative + review embeddings). • Clustering embeddings often *reduced* performance for collaborative-only methods (e.g., LightGCN, KGCL). • UserKNN excelled in HIT@50 and HIT@100, showing value of simple CF when ratings are sufficient.
Amazon Book (sparser)	• KGCL with both clustering and review embeddings often top performer (e.g., KGCL-DP-CC-R-4C). • Clustering alone less effective, but *clustering + reviews* gave large boosts. • Simpler methods (e.g., Spectral Clustering) surprisingly competitive when augmented with reviews.
Yelp (medium sparsity)	• AdaGCL best for smaller K (5, 10), KGCL best for larger K (50, 100). • Review embeddings valuable, especially for smaller lists; clustering impact minimal. • Top spots often held by baseline AdaGCL/KGCL without clustering.
Cross-dataset trends	• Review embeddings *consistently* improve performance, sometimes more than clustering. • Clustering helps mainly on sparse datasets when combined with reviews; can hurt in dense datasets. • Graph structure (knowledge graphs) crucial for sparse datasets, less so for dense ones.

between the methods are smaller but the impact of the augmentation with review and cluster centroid embeddings is larger. We summarize our results in Table 7.

In the future, it is likely worth further considering additional datasets with different levels of sparsity. Similarly, it would be good to run further experiments with diversity, novelty, and other measures to see the impact of considered components there. It would be also interesting to create knowledge graphs for other datasets to see whether our findings can be confirmed on additional datasets.

ACKNOWLEDGMENTS

This work is supported by a grant from the Carlsberg Foundation by grant agreement nr. CF21-0073.

REFERENCES

[1] P. Bellini, L. A. I. Palesi, P. Nesi, and G. Pantaleo. Multi clustering recommendation system for fashion retail. *Multim. Tools Appl.*, 82(7):9989–10016, 2023.

[2] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

[3] A. H. Brams, A. L. Jakobsen, T. E. Jendal, M. Lissandrini, P. Dolog, and K. Hose. Mindreader: Recommendation over knowledge graph entities with explicit user ratings. In *ACM CIKM 2020*, page 2975–2982. ACM, 2020.

[4] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, 2015.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[6] Y. Chen, Z. Liu, J. Li, J. McAuley, and C. Xiong. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2172–2182, New York, NY, USA, 2022. Association for Computing Machinery.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] P. A. B. Dezfouli, S. Momtazi, and M. Dehghan. Deep neural review text interaction for recommendation systems. *Applied Soft Computing*, 100, 2021.

[9] P. Dolog, Y. Sadikaj, Y. Velaj, A. Stephan, B. Roth, and C. Plant. The impact of cluster centroid and text review embeddings on recommendation methods. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 589–592, New York, NY, USA, 2024. Association for Computing Machinery.

[10] M. D. Ekstrand. Lenskit for python: Next-generation software for recommender systems experiments. In *ACM CIKM 2020*, Oct 2020.

[11] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He, and Y. Li. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Recomm. Syst.*, 1(1), Mar. 2023.

[12] W. Guo, R. Su, R. Tan, H. Guo, Y. Zhang, Z. Liu, R. Tang, and X. He. Dual graph enhanced embedding neural network for ctr prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 496–504, New York, NY, USA, 2021. Association for Computing Machinery.

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

[14] E. Hasan, M. Rahman, C. Ding, J. X. Huang, and S. Raza. Review-based recommender systems: A survey of approaches, challenges and future perspectives, 2024.

[15] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 507–517, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.

[16] W. He, G. Sun, J. Lu, and X. S. Fang. Candidate-aware graph contrastive learning for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 1670–1679, New York, NY, USA, 2023. Association for Computing Machinery.

[17] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *ACM SIGIR 2020*, page 639–648, 2020.

[18] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[19] Y. Jiang, C. Huang, and L. Huang. Adaptive graph contrastive learning for recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 4252–4261, New York, NY, USA, 2023. Association for Computing Machinery.

[20] T. N. Kipf and M. Welling. Variational graph auto-encoders. In *arXiv preprint arXiv:1611.07308*, 2016.

[21] M. Leginus, P. Dolog, and V. Zemaitis. Improving tensor based recommenders with clustering. In *UMAP 2012*, pages 151–163. Springer, 2012.

[22] C.-L. Li, A. G. Buyuktur, D. K. Hutchful, N. B. Sant, and S. K. Nainwal. Portalis: using competitive online interactions to support aid initiatives for the homeless. In *CHI '08 extended abstracts on Human factors in computing systems*, pages 3873–3878, New York, NY, USA, 2008. ACM.

[23] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 539–548, New York, NY, USA, 2019. Association for Computing Machinery.

[24] D. Liu, J. Li, B. Du, J. Chang, and R. Gao. Daml: Dual attention mutual learning between ratings and reviews for item recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 344–352, New York, NY, USA, 2019. Association for Computing Machinery.

[25] Q. Liu, N. Chen, T. Sakai, and X.-M. Wu. Once: Boosting content-based recommendation with both open- and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 452–461, New York, NY, USA, 2024. Association for Computing Machinery.

[26] Y. Liu, S. Zhu, J. Xia, Y. Ma, J. Ma, X. Liu, S. Yu, K. Zhang, and W. Zhong. End-to-end learnable clustering for intent learning in recommendation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 5913–5949. Curran Associates, Inc., 2024.

[27] H. Lyu, S. Jiang, H. Zeng, Y. Xia, Q. Wang, S. Zhang, R. Chen, C. Leung, J. Tang, and J. Luo. LLM-rec: Personalized recommendation via prompting large language models. In K. Duh, H. Gomez, and S. Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 583–612, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[28] J. Ma, C. Zhou, H. Yang, P. Cui, X. Wang, and W. Zhu. Disentangled self-supervision in sequential recommenders. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 483–491, New York, NY, USA, 2020. Association for Computing Machinery.

[29] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *ACL 2011*, pages 142–150. ACL, jun 2011.

[30] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.

[31] K. Mei and Y. Zhang. Lightlm: A lightweight deep and narrow language model for generative recommendation, 2023.

[32] P. Nema, A. Karatzoglou, and F. Radlinski. Disentangling preference representations for recommendation critiquing with ß-vae. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 1356–1365, New York, NY, USA, 2021. Association for Computing Machinery.

[33] A. N. Nikolakopoulos, X. Ning, C. Desrosiers, and G. Karypis. Trust your neighbors: A comprehensive survey of neighborhood-based methods for recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 39–89. Springer US, New York, NY, 2022.

[34] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. In *arXiv preprint arXiv:1807.03748*, 2018.

[35] X. Qin, H. Yuan, P. Zhao, G. Liu, F. Zhuang, and V. S. Sheng. Intent contrastive learning with cross subsequences for sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 548–556, New York, NY, USA, 2024. Association for Computing Machinery.

[36] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP 2019*. ACL, 11 2019.

[37] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI 2009*, page 452–461, 2009.

[38] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *ACM CSCW 1994*, 1994.

[39] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20:53–65, 1987.

[40] Y. Sadikaj, Y. Velaj, S. Behzadi, and C. Plant. Spectral clustering of attributed multi-relational graphs. In *ACM SIGKDD 2021*, page 1431–1440, 2021.

[41] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *ACM RecSys 2008*, page 259–266, 2008.

[42] R. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

[43] N. Tran and H. W. Lauw. Learning multi-faceted prototypical user interests. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[44] C.-F. Tsai and C. Hung. Cluster ensembles in collaborative filtering recommendation. *Applied Soft Computing*, 12(4):1417–1425, 2012.

[45] U. von Luxburg. A tutorial on spectral clustering. *Stat. Comput.*, 17(4):395–416, 2007.

[46] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 950–958, New York, NY, USA, 2019. Association for Computing Machinery.

[47] W. Wei, C. Huang, L. Xia, and C. Zhang. Multi-modal self-supervised learning for recommendation. In *ACM WWW 2023*, pages 790–800, 2023.

[48] Y. Wei, Y. Xu, L. Zhu, J. Ma, and C. Peng. Multi-level cross-modal contrastive learning for review-aware recommendation. *Expert Systems with Applications*, 247:123341, 2024.

[49] 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. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *EMNLP 2020: System Demonstrations*, pages 38–45, Online, Oct. 2020. ACL.

[50] C. Wu, F. Wu, T. Qi, and Y. Huang. Userbert: Pre-training user model with contrastive self-supervision. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2087–2092, New York, NY, USA, 2022. Association for Computing Machinery.

[51] Y. Yang, C. Huang, L. Xia, and C. Li. Knowledge graph contrastive learning for recommendation. In *ACM SIGIR 2022*, page 1434–1443, 2022.