

## Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

DocuSigned by:  
Signature: *Sergey Volokhin*  
DCFB1C1885EA47E...

Sergey Volokhin  
Name

7/30/2024 | 12:13 PM EDT  
Date

**Title**            Improving User and Item Representation for Recommender Systems

**Author**            Sergey Volokhin

**Degree**            Doctor of Philosophy

**Program**          Computer Science


**Approved by the Committee**

DocuSigned by:  
  
C81F95707F73454...

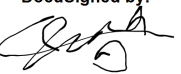
Eugene Agichtein  
*Advisor*

DocuSigned by:  
  
434F513F55B2437...

Filip Radlinski  
*Committee Member*

DocuSigned by:  
  
34E5DE72DFD0456...

Joyce Ho  
*Committee Member*

DocuSigned by:  
  
C5A3D3420EBA406...

Jinho D. Choi  
*Committee Member*

*Committee Member*

*Committee Member*

**Accepted by the Laney Graduate School:**

---

Kimberly Jacob Arriola, Ph.D, MPH  
Dean, James T. Laney Graduate School

---

Date

Improving User and Item Representation  
for Recommender Systems with Textual Data

By

Sergey Volokhin

Advisor: Eugene Agichtein, PhD

An abstract of

A dissertation submitted to the Faculty of the  
James T. Laney School of Graduate Studies of Emory University  
in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy  
in Computer Science  
2024

## Abstract

Improving User and Item Representation  
for Recommender Systems with Textual Data  
By Sergey Volokhin

Conversational Recommender Systems (CRS) aim to provide personalized and contextualized recommendations through natural language conversations with users. This dissertation capitalizes on the recent developments in conversational interfaces to advance the field of Recommender Systems in several directions. Specifically, I address the user and item representation and recommendation explainability problems in recommender systems.

First, I investigate several approaches for improving user representation. One approach is to map conversational users to reviewers with more data, using semantic similarity between the conversation and the texts of reviews. Another approach is to extract the items the user has mentioned, the sentiment the user has expressed towards them, and what specifically the user said about them, and use that information to represent the user.

Second, I investigate improving user and item representations by leveraging textual information. I develop methods to incorporate textual features such as item descriptions into the user-item interaction graph, which introduce additional semantic and behavioral information unavailable from the purely topological structure of the interaction graph. I also investigate whether the knowledge learned by LLMs during pretraining can be leveraged to improve the user and item representations by generating new textual features about the users and items.

Third, I investigate ways to improve explainability. One approach to enhance the explainability and transparency of CRS is to generate justifications. However, existing methods, such as rule-based and template-based methods, have limitations. In this work, I develop an extractive method using a corpus of reviews to identify relevant information for generating concise and coherent justifications.

The **research questions** I am tackling are:

1. How to infer and represent *user preferences* during a conversation with the system?
2. How to better represent users and items using structured and unstructured knowledge for improving the quality of recommendations?
3. How to improve the explainability of conversational recommendations?

This thesis improves the effectiveness of conversational recommender systems and advances the state-of-the-art in the field by introducing novel approaches for user and item representation for improving conversational recommendation systems.

# Improving User and Item Representation for Recommender Systems with Textual Data

**Sergey Volokhin**

Computer Science Department

Emory University

svolokh@emory.edu

Advisor: Eugene Agichtein, Ph.D.

August 9, 2024

## Abstract

Conversational Recommender Systems (CRS) aim to provide personalized and contextualized recommendations through natural language conversations with users. This dissertation capitalizes on the recent developments in conversational interfaces to advance the field of Recommender Systems in several directions. Specifically, I address the user and item representation and recommendation explainability problems in recommender systems.

First, I investigate several approaches for improving user representation. One approach is to map conversational users to reviewers with more data, using semantic similarity between the conversation and the texts of reviews. Another approach is to extract the items the user has mentioned, the sentiment the user has expressed towards them, and what specifically the user said about them, and use that information to represent the user.

Second, I investigate improving user and item representations by leveraging textual information. I develop methods to incorporate textual features such as item descriptions into the user-item interaction graph, which introduce additional semantic and behavioral information unavailable from the purely topological structure of the interaction graph. I also investigate whether the knowledge learned by LLMs during pretraining can be leveraged to improve the user and item representations by generating new textual features about the users and items.

Third, I investigate ways to improve explainability. One approach to enhance the explainability and transparency of CRS is to generate justifications. However, existing methods, such as rule-based and template-based methods, have limitations. In this work, I develop an extractive method using a corpus of reviews to identify relevant information for generating concise and coherent justifications.

The **research questions** I am tackling are:

1. How to infer and represent *user preferences* during a conversation with the system?
2. How to better represent users and items using structured and unstructured knowledge for improving the quality of recommendations?
3. How to improve the explainability of conversational recommendations?

This thesis improves the effectiveness of conversational recommender systems and advances the state-of-the-art in the field by introducing novel approaches for user and item representation for improving conversational recommendation systems.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Research Questions . . . . .	5
1.2	Contributions . . . . .	5
<b>2</b>	<b>Related Work</b>	<b>7</b>
2.1	Recommender Systems Overview . . . . .	7
2.1.1	Content-Based Filtering . . . . .	7
2.1.2	Collaborative Filtering . . . . .	7
2.1.3	Hybrid Methods . . . . .	8
2.1.4	Neural Methods . . . . .	9
2.1.5	Graph-based Methods . . . . .	9
2.1.6	LLM-based Methods . . . . .	10
2.2	Conversational Recommender Systems . . . . .	11
2.2.1	Types of Conversational Recommender Systems . . . . .	11
2.2.2	Conversational User Representation for Recommendations . . . . .	12
2.3	Explainable Recommendations . . . . .	12
2.3.1	Review-based Methods . . . . .	13
2.3.2	Deep Learning Methods . . . . .	13
2.3.3	Context-Aware and Conversational Methods . . . . .	13
2.3.4	LLM-Based Methods . . . . .	14
2.3.5	Evaluation and Impact . . . . .	14
2.4	Knowledge Injection for Recommender Systems . . . . .	14
2.5	Data Augmentation Using Foundational Language Models . . . . .	15
2.5.1	Conversational Recommender Systems . . . . .	15
2.5.2	Traditional Recommender Systems . . . . .	16
2.6	Summary . . . . .	17
<b>3</b>	<b>User Preference Representation in Conversational Recommendations</b>	<b>18</b>
3.1	Method Overview . . . . .	19
3.1.1	Collaborative Filtering . . . . .	19
3.1.2	Domain Adaptation . . . . .	19
3.1.3	Model . . . . .	20
3.2	Data . . . . .	21
3.2.1	<i>MovieSent</i> : Sentiment Elicitation Dataset . . . . .	21
3.2.2	Reviews dataset . . . . .	21
3.3	Metrics . . . . .	21
3.4	Experiments . . . . .	22
3.4.1	Baselines . . . . .	22

3.4.2	ConvExtr . . . . .	22
3.4.3	Evaluation Procedure . . . . .	23
3.5	Results . . . . .	23
3.6	Summary . . . . .	23
3.7	Limitations . . . . .	24
<b>4</b>	<b>Knowledge Injection for Better Recommendations</b>	<b>25</b>
4.1	Method Overview . . . . .	26
4.1.1	Training LightGCN . . . . .	26
4.1.2	User and Item Representation . . . . .	27
4.2	Data . . . . .	27
4.3	Metrics . . . . .	28
4.4	Experiments . . . . .	28
4.4.1	Baselines . . . . .	28
4.4.2	TextGCN . . . . .	29
4.5	Results . . . . .	30
4.6	Summary . . . . .	32
4.7	Limitations . . . . .	32
<b>5</b>	<b>Enhancing Recommender Systems with LLM-Driven User and Item Insights</b>	<b>33</b>
5.1	Methodology . . . . .	35
5.1.1	Model Training . . . . .	36
5.1.2	Implementation Details . . . . .	36
5.1.3	Alternative Backbones . . . . .	37
5.2	Data . . . . .	38
5.2.1	Source . . . . .	38
5.2.2	Preprocessing . . . . .	38
5.3	Experiments . . . . .	39
5.3.1	Feature Generation . . . . .	39
5.4	Results and Analysis . . . . .	40
5.4.1	User Features . . . . .	42
5.4.2	Item Features . . . . .	42
5.4.3	LightGCN vs MMSSL . . . . .	43
5.4.4	Feature Interactions . . . . .	43
5.4.5	Feature Importances . . . . .	44
5.5	Summary . . . . .	44
5.6	Limitations . . . . .	45
<b>6</b>	<b>Using Textual Data for Explanation Construction</b>	<b>46</b>
6.1	Method Overview . . . . .	47
6.1.1	Elementary Discourse Units (step 1) . . . . .	47
6.1.2	Generate and Select Candidates (step 4) . . . . .	48
6.2	Data . . . . .	48
6.2.1	Conversation Corpus . . . . .	48



6.2.2	Reviews Corpus . . . . .	48
6.3	Metrics . . . . .	49
6.4	Evaluation . . . . .	49
6.4.1	Procedure . . . . .	49
6.4.2	Baselines . . . . .	49
6.5	Experiments . . . . .	50
6.6	Results . . . . .	51
6.7	Summary . . . . .	51
6.8	Limitations . . . . .	52
<b>7</b>	<b>Summary and Discussion</b>	<b>53</b>
7.1	RQ1 . . . . .	53
7.2	RQ2 . . . . .	53
7.2.1	RQ2(a,b) . . . . .	53
7.2.2	RQ2(c) . . . . .	54
7.3	RQ3 . . . . .	54
7.4	Ethical Considerations . . . . .	55
7.4.1	Bias and Fairness . . . . .	55
7.4.2	Privacy . . . . .	55
7.4.3	Transparency . . . . .	56
7.4.4	Factuality in CONJURE . . . . .	57
7.5	Conclusions . . . . .	57
<b>A</b>	<b>Prompts for Feature Generation</b>	<b>70</b>

# 1. Introduction

The widespread use of recommender systems is a common feature of daily life, and the growing interest in conversational agents presents fresh research opportunities.

A massive shift in the search paradigm is underway: Microsoft has integrated an LLM into their search, and Google has followed suit. More people are using free-form textual generation tools, which are more intuitive and easier to use than traditional approaches that return a ranked list of pages and let the users find the answer themselves.

Recommendation systems are very similar to search engines, and it is reasonable to assume that the same technologies will gain comparable popularity and usage. Therefore, conversational recommendations are more relevant and important now than ever.

Figure 1.1 shows an example conversation between a user and a conversational recommender system. The system tries to determine user preferences and come up with a recommendation. The user can then accept the recommendation or ask for another one. The system can also ask for more information from the user to improve the recommendation. They exchange utterances until the system comes up with a final recommendation.

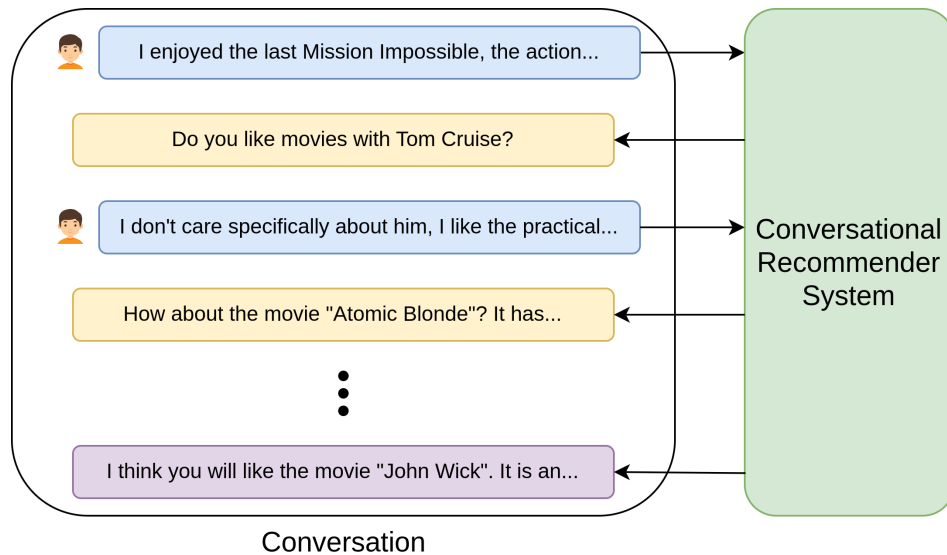


Figure 1.1: Example of a dialog flow for a conversational recommender system. The user utterances are fed into the system, generating initial recommendations or clarification questions. The user can answer the questions and iteratively give feedback on the recommendation, and the system will generate a new recommendation based on the feedback

## 1.1 Research Questions

In this work, I address some existing open research questions in conversational recommender systems (CRS), like how to represent the user and items, estimate the user preference with utterances, and improve general satisfaction with the system by introducing conversation-based explanations for recommended items.

More formally, the research questions are:

1. How to represent user preferences using conversation?  
More specifically: Can we represent user preference towards items mentioned and not mentioned in the conversation?
2. How to better represent users and items using structured and unstructured knowledge to improve the quality of recommendations? The specific sub-questions are:
  - (a) How should the knowledge be injected into the Knowledge Graph?
  - (b) Which information about the users and items is most useful for recommendations?
  - (c) Can LLMs be used to improve user and item representations?
3. How can we improve the explainability of conversational recommendations?  
More specifically: Which particular aspects of explanations can be improved with user-generated textual data (e.g., reviews)?

The architecture of CRS is illustrated in Figure 1.2. The system takes the conversational context as input and any other information about the target user, such as their previous reviews or profiles. Those inputs are fed into a *user representation module*, which returns the user representation as an embedding vector (top green box). At the same time, the system also feeds all available information about the items into the *item representation module*, which returns the item representation as an embedding vector (bottom green box). The user and item representations are then fed into the *recommendation module*, which returns the recommendation item (blue box). The item and the user and item representations are then fed into the *explanation module*, which returns the explanation for the recommendation (blue box). Finally, the recommendation and the explanation are returned to the user as a complete conversational recommendation (final blue box).

## 1.2 Contributions

My contributions are as follows:

- Create a knowledge-aware CRS that combines conversational context with external knowledge, such as movie reviews, to predict users' ratings for unseen movies (§3);

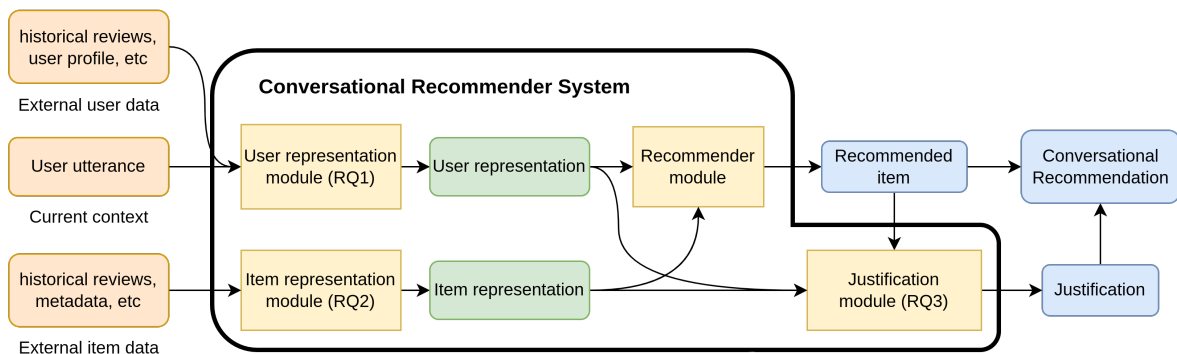


Figure 1.2: How proposed research questions connect to form the complete recommendation pipeline. Orange squares are inputs. Yellow squares are all the recommender system modules. Blue squares are the outputs.

- Develop a way to augment interaction-based Graph Recommender Systems with textual information for improved node representation and introduce a simple and general approach for integrating both graph and textual representations of users and items (§4);
- Design a technique to further augment the textual information with LLM-generated features to improve the user and item representations for recommendations (§5);
- Introduce a method for using a conversation in place of the current user profile to generate explanations for recommendations (§6);

## 2. Related Work

This chapter describes the related work to place my contributions into context.

### 2.1 Recommender Systems Overview

This section briefly describes the traditional recommender systems methods I build upon.

#### 2.1.1 Content-Based Filtering

A content-based filtering recommender system is a type of recommendation system that generates suggestions based on user profiles and the content of items. Users' preferences are typically embedded using a profile built from the aggregation of information on the content of items the user has interacted with, for instance, the genres of movies a user has watched or the topics of articles they have read. These user profiles can then be represented in a multidimensional space where each dimension corresponds to a unique characteristic or feature of the items. The similarity between users or items is generally computed using measures like cosine similarity or Pearson correlation in the multidimensional feature space. For example, two users are considered similar if their profiles have high cosine similarity, indicating they have interacted with similar items. Likewise, two items are seen as similar if they share many features, suggesting they could appeal to the same type of users. The system then recommends items similar to the ones the user has already liked or interacted with, providing a personalized experience [57, 71, 82].

#### 2.1.2 Collaborative Filtering

The main difference between Collaborative Filtering (CF) [36] and Content-based filtering is that CF does not need the features of the items to be given. A feature vector or embedding describes every user and item. The algorithms for collaborative filtering can be divided into two main categories: model-based and memory-based.

##### Memory-based algorithms

Memory-based algorithms use the entire user-item interaction matrix to calculate the similarity between users or items. There is no dimensionality reduction or model fitting. Some potential drawbacks of memory-based CF include scalability and sensitivity to data sparseness [62].

Memory-based algorithms can be divided into two main types.

- **User-based:** recommends items by finding users with past behavior similar to the target user and suggesting items these similar users liked.
- **Item-based:** recommends items by identifying similar items (often rated similarly by users) to those the target user has liked in the past.

One popular approach is neighborhood-based CF or clustering. Clustering algorithms group users or items into clusters based on their similarities. The system then recommends items to users based on the users' preferences in the same cluster by calculating a weighted sum of the neighbors' scores. Examples of clustering algorithms include K-means [40], DBSCAN [31], and Hierarchical clustering [52].

## Model-based algorithms

Model-based algorithms use machine learning techniques to learn how a specific user or an item behaves. The large interaction matrix is compressed using dimensional reduction or clustering algorithms.

The most common model-based approach is matrix factorization (MF) [56]. MF is a model class that decomposes the user-item interaction matrix into the product of two lower-dimensional matrices. The first matrix represents the users, and the second represents the items. The elements of the resulting matrices are latent factors that represent the user's and item's characteristics. The dot product of the two matrices approximates the user-item interaction matrix. The resulting matrix can predict a user's rating for an item they have not rated yet. The predicted ratings can be used to recommend items to users. Examples of MF algorithms include Singular Value Decomposition (SVD) [56], Non-negative Matrix Factorization (NMF) [61], and Probabilistic Matrix Factorization (PMF) [94].

### 2.1.3 Hybrid Methods

Hybrid recommendation systems [14] can produce outputs that outperform single-component systems by combining multiple techniques of different types, such as mixing content-based and collaborative filtering methods [105].

The most popular techniques for hybrid recommendation systems include the following:

- **Weighted:** The scores of different recommendation techniques are combined using a weighted average.
- **Switching:** The system switches between different recommendation techniques based on user profiles.
- **Cascade:** selects a candidate entirely with the main recommendation and uses the other recommendation to refine product or item scores.
- **Mixed:** The scores of different recommendation techniques are combined using a linear or non-linear function.
- **Feature Combination:** The features of different recommendation techniques are combined using a linear or non-linear function.

- **Feature Augmentation:** The features of one recommendation technique augment the features of another.

Hybrid methods combine the benefits of the constituent approaches while mitigating their flaws. For instance, CF methods suffer from new-item problems, i.e., they cannot recommend items with no ratings. This allows content-based approaches to be more consistent since the prediction for new items is based on their description (features) that are typically readily available [57]. Hence, hybrid methods that include both of them can overcome this problem.

### 2.1.4 Neural Methods

Traditional collaborative filtering methods, such as matrix factorization, are effective but have certain limitations. They can struggle with sparse data or complex, non-linear user-item interactions. Neural Collaborative Filtering [45] aims to address these limitations. It represents user-item interactions as low-dimensional dense vectors, which are then passed through a neural network to model the complex, non-linear relationships between users and items.

Another popular Neural Recommender System example is the Deep Learning Recommendation Model (DLRM) [80], which builds upon existing work on factorization machines [91] and deep neural networks [45]. DLRM uses embeddings to represent the categorical features of the users and items and passes continuous features through a multilayer perceptron (MLP), which will yield a dense representation of the same length as the embedding vectors. The embeddings are then passed through a series of fully connected layers to learn the latent representations of the users and items.

### 2.1.5 Graph-based Methods

While falling under the broader category of neural approaches, graph-based techniques warrant separate consideration due to their significant scope and impact. It is a recommender system class that uses graph neural networks (GNN) [96] to model the user-item interaction graph. The user-item interaction graph is a bipartite graph where the nodes are users and items, and the edges represent the interactions between users and items. The graph can be constructed from explicit user-item interactions, such as ratings, or implicit interactions, such as clicks or purchases. The graph can also be augmented with additional information about the users and items, such as user profiles and reviews or item descriptions and attributes. The GNN can then be used to learn the latent representations of the users and items. The learned representations can predict the likelihood of a user interacting with a given item, which can be used to recommend items to the user.

The most well-known graph-based recommender systems, including GCN [55], GraphSAGE [38], and GAT [107], were introduced in the pre-transformer era. These methods are based on the Graph Convolutional Network (GCN) [55] architecture. The GCN architecture is a type of GNN that uses a convolution to learn the latent representations of the nodes in the graph. The convolutional filter is applied to each node and its

neighbors. The resulting feature vectors are concatenated to form the output feature vector for the node. The output feature vector is then passed through a non-linear activation function to produce the final output feature vector for the node.

More recently, self-attention mechanism [63] and contrastive learning [63, 97, 102, 120] have been incorporated directly into graph models to identify the most important interactions and facilitate self-supervised learning.

I build upon existing work on graph-based recommender systems, such as LightGCN, in the proposed algorithm addressing RQ2, discussed in Section 4.

## 2.1.6 LLM-based Methods

Ranking and scoring methods are slowly but surely giving way to generative recommendations, where the recommendation is generated without the need to calculate each candidate’s ranking score one by one [65]. However, LLMs have interesting applications in scoring and ranking to improve recommender systems as well, not just as end-to-end generative recommender systems.

Li et al in [68] evaluate ChatGPT’s abilities on 5 different recsys-related tasks:

- rating prediction
- direct recommendation
- sequential recommendation
- review summary
- explanation generation

They try both zero- and few-shot settings, and their results suggest that “ChatGPT performs well in rating prediction but poorly in sequential and direct recommendation tasks”.

### In Context Learning

There have been many attempts at zero- and many-shot learning for LLMs in the recommendation domain.

Dai et al. in [25] propose a re-ranking method for LLMs, with 3 distinct capabilities: point-wise, pair-wise, and list-wise ranking.

He et al. in [46] decided to evaluate LLM’s performance in the conversational setting. They report several interesting findings:

- LLMs outperform fine-tuned CRS models in a zero-shot setting
- GPT-based LLMs possess better content/context knowledge than existing CRS
- LLMs generally possess weaker collaborative knowledge than existing CRS
- LLM recommendations suffer from popularity bias in CRS

In all, they conclude that LLMs are a promising direction for conversational recommendations.

Sanner et al. in [95] investigated LLM performance in the near cold-start context. They have discovered that zero- and few-shot strategies in LLMs provide competitive performance for pure language-based preferences (no item preferences) in the near cold-start case in comparison to item-based collaborative filtering methods.



## Fine-Tuning

Bao et al. in [6] developed a framework to fine-tune LLMs for recommendation task. It consists of 2 parts: alpaca tuning and recommendation tuning, where the former is used to enhance the model’s generalization ability and the latter to improve the recommendation performance. Proposed rec-tuning objective is a binary classification task, where the model is asked whether the user would like a proposed item or not. They have shown that fine-tuning LLMs can significantly improve the recommendation performance compared to untuned models, and exhibit strong cross-domain generalization abilities.

Liu et al in [69] propose to combine open- and closed-source LLMs for content-based recommendations. One of the steps in the framework is fine-tuning an open-source LLM to act as a content-encoder. This step alone shows the most significant improvements. Next, the closed-source LLM is used as data augments using a variety of prompts: for content summarization, user profiling, and content generator. Using this data during training further improves the recommendation performance. In all, the proposed approach shows substantial improvements.

Cao et al in [15] propose using several new pretraining objectives to align the LLMs with the recommendation task. The objectives include Masked Item Modeling, Masked Language Modeling (different from the original MLM [27]), and Bayesian Personalized Ranking. Furthermore, the data samples that the model is trained on are designed to be more informative than in existing studies. Results show that the proposed method effectively introduces recommendation knowledge, and outperforms both conventional and LLM-based baselines in retrieval.

## 2.2 Conversational Recommender Systems

Conversational Recommender Systems (CRS) are a particular type of Recommender Systems that acquire the user’s profile in an interactive manner through natural language, versus the traditional systems that are usually based on analysis of past user behavior [33, 73]. CRSs were developed to address the limitations of traditional recommender systems, such as the cold-start problem, the sparsity problem, and the lack of user feedback. They are more natural and intuitive for users to interact with, as they mimic how humans interact. However, that introduces new challenges, such as the need to understand natural language, extract the user’s intent, maintain the context and state of the conversation, etc. It also requires the system to be able to generate natural language responses that are relevant to the user’s request. Modern LLMs are well-suited for these tasks since they work with natural language and can generate human-like responses by design.

### 2.2.1 Types of Conversational Recommender Systems

CRSs can be broadly split into 3 categories: system-driven, user-driven, and mixed-initiative [50]. In system-driven methods, the users are typically asked about their preferences and then presented with an initial recommendation. Users can then use a set of pre-defined or dynamically determined critiques to refine their preferences further. In

user-driven systems, the system takes no proactive role. The resulting dialogue, therefore, consists of “user-asks, system-responds” pairs [135]. Such conversation patterns are typical for one-shot query-answering, search, and recommendation systems. The most popular approach, however, is the mixed-initiative approach, where the system takes a proactive role in the conversation. The user can also take the initiative and ask questions [50].

Figure 1.2 shows some components of a Conversational Recommender System in the context of this work.

## 2.2.2 Conversational User Representation for Recommendations

Establishing the new user’s preferences through a conversation to make an effective recommendation remains an open question. Most efforts in the conversational recommendations field focus on preference elicitation by asking user questions [131, 58]. However, most work establishes user preference through individual item attributes/-facets [100, 135].

Kostric et al. in [58] propose asking implicit questions based on item usage to establish user preferences. The questions are generated using a neural model based on the reviews about the item. The authors have shown that proposed method is effective at eliciting user preferences, but they have stopped short of actually using the user answers in a recommendation algorithm, and proposed that as future work.

In [89], a model called “Navigation-by-Preference” (n-by-p) is introduced, which is a recommendation made in a content-based way using unstructured item descriptions such as sets of keywords or tags. While the authors use items rather than attributes to establish user preferences, the proposed approach uses user-to-user filtering and user reviews instead of item-to-item filtering and item descriptions as the basis for the CF.

I explore a new approach to conversational recommendations by incorporating preferences of other external users with established preferences via shared discussed entities and the user’s sentiment towards them. It also addresses the resulting “cold start” problem. In this setting, users do not ask for recommendations directly but instead have a more natural conversation with a Wizard and receive recommendations based on this discussion. Previous approaches, such as the “hierarchy of recommendation goals” [54] and “narrative-driven recommendations” [9, 29], are not applicable under these conditions.

## 2.3 Explainable Recommendations

Explainable recommendation refers to personalized recommendation algorithms that address the problem of why – they not only provide users or system designers with recommendation results but also explanations to clarify why such items are recommended [136]. Providing justification or explanation for a recommendation has been shown to improve users’ experience with recommender systems, particularly by increasing confidence in the recommendations [35, 47, 98, 130].

I will now describe the specific types of explainable recsys relevant to my work

### 2.3.1 Review-based Methods

There has been a considerable recent effort to use user-generated review text to provide explanations or justifications for recommender systems, including Matrix [137] and Tensor [18] Factorization, Aspect Extraction [78, 81], Topic Modeling, Word Clouds [75, 125], and Graph-based methods [42].

These methods often require pre-existing history of product interactions, reviews, or even, specifically, justifications, like [23, 42, 60, 81, 122], limiting their applicability to new users.

### 2.3.2 Deep Learning Methods

Wu et al. in [122] propose a Deep Conversational Critiquing Framework that provides explanations based on *contextualized descriptive key phrases* and the recommended item. The key phrases consist of uni- and bi-grams, extracted beforehand from reviews using a custom algorithm. While they achieve high performance of up to 88% of relevant key phrases returned, they are not formed into a coherent sentence that can be used in a conversational setting, for example, returned by a voice assistant or chatbot. Here, I generate explanations that can be used naturally in conversation.

BERT probing was tried for Conversational Recommendations [84], and while the authors show their improvement for recommendations, they also show that the accuracy of probing is relatively low, with correct genres appearing in top-5 predictions for only 30-50% of cases.

Some systems use deep neural network models to generate text or fill in positive-sentiment templates (e.g., [23, 81, 85]), which may lead to inaccuracies.

### 2.3.3 Context-Aware and Conversational Methods

Balog et al. in [4] propose modeling user preferences based on sets of related items. The sets allow to generate meaningful sentences that can be used as explanations after reranking according to user preferences. User preferences are modeled as a likelihood estimation over items' tags. As a result, the proposed method is more transparent and scrutable than the previous methods.

Penha et al. in [85] propose a pairwise review-based explanation for voice search recommendations. This is a unique setup with its own challenges, such as lack of visual cues, user memory limitations and user fatigue. The proposed template-based method is applied to a pointwise setting (only one item is presented) and a pairwise setting (two items are presented). The results show that the pairwise setting is significantly more effective at Effectiveness and Transparency metrics.

### 2.3.4 LLM-Based Methods

It has been previously shown that LLMs can produce highly relevant and persuasive explanations for recommendations [113] due to their natural ability to generate texts and strong capabilities in logical and causal reasoning [115]. However, the evaluation was performed using the LLM that produced the explanations as a scorer instead of real humans. The prompt used for evaluation did not employ the best-known techniques of prompt tuning like CoT [119], Self-Consistency [114], or few-shot learning.

Since then, more rigorous work has been produced, based on extracted item aspects [88], continuous prompt learning [64], knowledge base alignment and personalized reasoning graph construction [115]. The latter is based on chained graph reasoning technique and has a self-verification step, which together solve the hallucination problem plaguing the LLMs. The knowledge base allows to inject any information, for example item metadata or user profiles.

### 2.3.5 Evaluation and Impact

Balog et al. in [3] measure how different explanation goals interact with each other, and propose a method of robustly measuring if explanation meets the goals. The authors find that all the goals are moderately correlated, while several pairs of goals are strongly correlated, and that the wording of the goals is critical for human quality assessment.

Furthermore, Balog et al. in [5] study how bias in the explanations can impact users' choices. They intentionally bias produced explanations towards or against certain aspects and measure the influence on user decisions. The explanations are either produced in an itemized format, comprising a list of extracted aspects, or in a fluent natural language format, by querying PaLM [20].

Lajewska et al. in [59] also ask the question of how the quality of explanations affects user-percieved usefulness. They have shown that users are insensitive to the completeness, factuality or fairness of the explanations, which signals that users are not able to identify the problems within the response without expert knowledge.

## 2.4 Knowledge Injection for Recommender Systems

Although new methods like Retrieval-Augmented Generation and Many-Shot prompting might not require a database of information about item, most legacy methods do, and the new methods could arguably also benefit from it. Such a database can contain item ratings, metadata that can be presented to the user (e.g., the genre of a movie or the director), community-provided tags, or extracted key phrases. These item attributes can serve as a basis for other computational tasks, e.g., to compute personalized recommendations, generate explanations, or determine which questions the user can be asked [50].

When using the additional information about items or users, the question arises: How can this information best be incorporated into the recommendation process?

Many systems that use additional information about entities or users augment the user-item graph with additional nodes. For instance, TGCN [16] includes a third class

of nodes called *tags*, which encode additional structured metadata. KGAT [112] includes categorical entities in the graph connected to items (for example, actor or director entities are connected to movie items).

Mei et al. in [76] connect additional entity nodes directly to users instead of constructing an interaction graph with user-entity and user-item edges. KCAN [106] also attempts to encode knowledge graphs alongside the user-item interaction graph. However, it uses more complex methods to incorporate that information into user and item representations; it achieves mostly minor improvements over KGAT.

More recent models are able to directly introduce multimodal information, including text, audio, and image, about the items and users into the models by performing graph fusion over different modality graphs to discern latent structures [133] or by training separate copies of the graph network for each modality and combining the representations [102, 120].

Zhu et al. in [141] propose a generative recommender system called CLLM4REC. It extends the vocabulary of a pretrained LLM with user and item ID tokens, which is then fine-tuned using a recommendation-oriented strategy on the interaction and knowledge graphs and converted into sentences in a natural language.

## 2.5 Data Augmentation Using Foundational Language Models

This section briefly describes most the popular ways to use Foundational Language Models for data augmentation and discusses the scarcity of Conversational Recommendations datasets.

### 2.5.1 Conversational Recommender Systems

Despite the rise in popularity, the CRSs still lack the large datasets that traditional recommender systems have had for years, like MovieLens [39] (up to 20M samples) or Netflix Prize [8] (100M samples). The biggest dataset of Conversational Recommendations generated by real humans I know of is OpenDialKG [77], with 12k conversations in the recommendation domain. Others are even smaller: ReDIAL [67] and TG\_ReDial [140] have 10k conversations each, GoRecDial [53] has 9k conversations, DuRecDial2.0 [70] has 8.2k (in English), and INSPIRED [41] has 1k. The Reddit-Movie dataset [46] has 643k conversations. However, it is built from the user comments on Reddit, which means most of the “conversations” are very short, with 61% having only one turn and 96% having less than 3 turns. There are many reasons for such scarcity: a single sample from the CRS dataset is an entire conversation with multiple turns and items mentioned, while a sample for MovieLens is just one number.

Data augmentation is a promising approach that became viable only recently with the development of Generative AI. The recent explosion in interest towards Foundational Language Models caused a chain reaction of open source creating better datasets, which help create better models, which help create even better datasets, and so on. Some examples include Vicuna [19], fine-tuned on real people’s conversations with ChatGPT

collected via ShareGPT; Alpaca [103], fine-tuned on a self-instruct-style [116] dataset generated by `text-davinci-003` model; and GPT4All [1], fine-tuned on a curated set of prompt-response pairs, generated using GPT-turbo-3.5 model. These models fine-tuned existing models using data created by another SOTA model.

Vicuna and GPT4All datasets were generated semi-automatically, so a human is still needed to provide the questions/prompts/other side of the conversation, which is much better than using 2 humans but can still be costly in terms of time and money. The dataset for Alpaca was collected completely automatically, where LLM generates both the instructions (prompts) and the responses (answers), using a handful of human-written samples as few-shot examples. However, it consists of only one type of interaction - the instruction data - and does not translate into conversation-style interactions.

Another synthetic dataset of conversations is LLM-REDIAL [2], which has not been officially released as of writing this thesis. It consists of 46.9k conversations, generated using templates by filling them with real users' information extracted from reviews by ChatGPT-3.5-turbo. The templates are split by goals, and the dialog structure is varied to ensure diversity.

Automatic high-quality conversation generation can provide researchers with data across any domain faster and more cost-effectively than human input. Moreover, this method can address inherent issues in real conversation datasets, such as class or length imbalance, by offering fine control over the attributes of the dialogue and preferences expressed in it. Finally, LLMs can cover a broad range of topics, providing an advantage over humans, who might lack expertise in specific domains and base recommendations on personal experiences only.

## 2.5.2 Traditional Recommender Systems

LLMs are also extensively used for generating synthetic training data [69, 79], user [34, 69, 121, 132] and item [69, 132] representations, and more.

More specifically, Liu et al. in [69] proposed to use LLMs as User Profiler by inferring the geographical region the user is in and topics the user is interested in. They then fuse the topics and regions of interest into one vector and use it as an "interest-aware user vector representation".

Wei et al. in [121] also use LLMs to generate "augmented side information", or more plainly, user and item attributes, which are then used to augment the user and item representations. They query the LLM to produce user profiles with their inferred demographic information: age, gender, country, and language, as well as preferences like genres and directors they like and dislike. Similarly, they infer item attributes like director, country and language.

Lui et al. in [69] also use LLMs to do free-form summarization for items, providing the title, abstract, and category as input to the LLM.

Zhang et al. in [132] split user profile into 2 components: social traits and unique tastes, they further break down social traits into 3 categories: activity, conformity, and diversity, and provide formulas for each. Users' preferences are encoded by taking 25 random items from their viewing history and splitting them into "liked" and "disliked", and then using an LLM to generate a summary of tastes.

Wu et al. in [123] propose a data distillation technique using LLM to generate a small dataset of high-quality samples, with the goal of achieving comparable quality to the original dataset with a fraction of the data. One benefit of the proposed approach is that it generates discrete text, which is easier to interpret and analyze than continuous representations. Another distinction from related methods is that the proposed method is training-free, and utilizes the extensive pre-training of the LLM. The authors report they have achieved 97% of the original dataset’s performance with only 5% of the samples.

Mysore et al. in [79] tackle the narrative-driven recommendations, where user’s preferences are presented with verbose descriptions and their context. The method utilizes LLM to augment training data with synthetic queries generated from historical user interactions.

Chen et al. in [17] propose a framework for generating data to improve fairness in recommendation systems. It is based on assumption that augmented data should be balanced between user groups, so that the system could learn preferences of all the groups. The authors generate both fake positive and negative data to equalize the data distribution. The results show that the proposed method improves both fairness and accuracy metrics on a variety of datasets and compared against several recent fairness-aware baselines.

## 2.6 Summary

Although extensive research methods exist in the established field of Recommender Systems, its emerging sub-discipline, Conversational Recommender Systems, still has many open research questions and challenges. At the same time, emerging tools like Foundational Language Models can enhance existing methods and create new ones.

The questions I will address in this work are: How to represent conversational user preferences in §3, how to represent the users and items using textual data in §4, how to improve user and item representations using LLMs in §5, and how to generate explanations for recommendations in §6.

### 3. User Preference Representation in Conversational Recommendations

In this chapter, I describe my work to address RQ1: “How to represent user preferences using conversation?”. More specifically, Can we represent user preference towards items mentioned and not mentioned in the conversation? The content of this chapter is based on the paper “You Sound Like Someone Who Watches Drama Movies: Towards Predicting Movie Preferences from Conversational Interactions” [110]<sup>1</sup>, published at NAACL’21.

Establishing the new user’s preferences through a conversation in order to make an effective recommendation remains an open question. This work explores one promising direction for conversational recommendation: mapping a conversational user, for whom there is limited or no data available, to most similar external reviewers, whose preferences are known, by representing the conversation as a user’s interest vector and adapting collaborative filtering techniques to estimate the current user’s preferences for new movies.

The proposed method, called *ConvExtr* (Conversational Collaborative Filtering using External Data), 1) infers a user’s sentiment towards an entity from the conversation context, and 2) transforms the ratings of “similar” external reviewers to predict the current user’s preferences. These steps are implemented by adapting contextual sentiment prediction techniques and domain adaptation, respectively. To evaluate this method, a finely annotated dataset of movie recommendation conversations, called *MovieSent*, is developed and made available. It is based on a previously released in [87] dataset of conversational preference elicitation. The results demonstrate that *ConvExtr* can improve the accuracy of predicting users’ ratings for new movies by exploiting conversation content and external data.

The formal statement of the problem is: given a prefix of  $k$  turns of conversation and mentions of  $m$  movies, we aim to predict the rating for the next movie  $m + 1$  to be mentioned in the conversation.

In the experiments described here, the value of  $m$  is set to 2, which approximates the average number of movies mentioned in a conversation with a voice assistant, but that could be extended. In that setting, the user’s preferences are estimated based on the first two movies mentioned in the conversation and predicted their rating for a third (unseen) movie.

---

<sup>1</sup>The code for the model is available at <https://www.github.com/sergey-volokhin/conversational-movies>



## 3.1 Method Overview

This section provides an overview of the proposed method and how the features were calculated. The complete model pipeline can be seen in Figure 3.1. Given a conversation about movies, first, user sentiment towards the mentioned movies is estimated and then used as input to a CF model to predict the rating for an unseen movie. The CF model uses a large set of external critics’ ratings and reviews, which should include critics similar to the current user. The final model, a Gradient Boosted Decision Tree, uses 3 main inputs: (1) CF predictions for the unseen movie; (2) similarity between the conversation user and critics; (3) similarity between the conversation and the movies’ metadata, to predict a score that the user would give each movie.

Specifically, we first construct the conversation representation and use pre-trained BERT to embed it using the sequence-encoder functionality of the model[128], which gives us one vector per conversation. Then we infer the fine-grained user sentiment for the movies discussed in the conversation using a Random Forest model, trained on the labeled dataset MovieSent (described in 3.2). Since this is not the focus of our work, we evaluated the prediction performance on a development set against manually annotated sentiment labels, resulting in an RMSE of 0.88 (mean over 10 tries, with std 0.06), which was sufficient for the current work.

The next step expands on a CF model (described in Section 3.1.1), constructed from an external reviews corpus, and predicts the score for the unseen movie. To make this prediction, we identify reviewers similar to the current conversation user via the similarity of their reviews to the current conversation text. We then calculate BERT-based sentence embeddings for all reviews of those critics and represent each critic as a centroid of their review vectors. Finally, we use the similarity between the conversation- and critics’ representations to transform the critics’ scores to predict the conversational users’ ratings.

### 3.1.1 Collaborative Filtering

The conversational user’s ratings are required to apply collaborative filtering (CF) algorithms. Since only the user’s conversational text is available, A sentiment analysis model is trained to infer the user’s sentiment toward the movie mentioned in the conversation. Then, these sentiment scores were converted to ratings and were used in traditional Collaborative Filtering (CF) algorithms to estimate the user’s sentiment toward a new movie. The basis for the CF was extensive external data of movie reviews and ratings, which were scraped from a popular website, RottenTomatoes (RT)<sup>2</sup>.

### 3.1.2 Domain Adaptation

I argue that the Conversational Users are sufficiently different from the Professional Critics whose reviews were used to build the CF matrix. To correctly predict the score for the users, additional features are introduced, with the expectation that they would

---

<sup>2</sup><http://rottentomatoes.com>

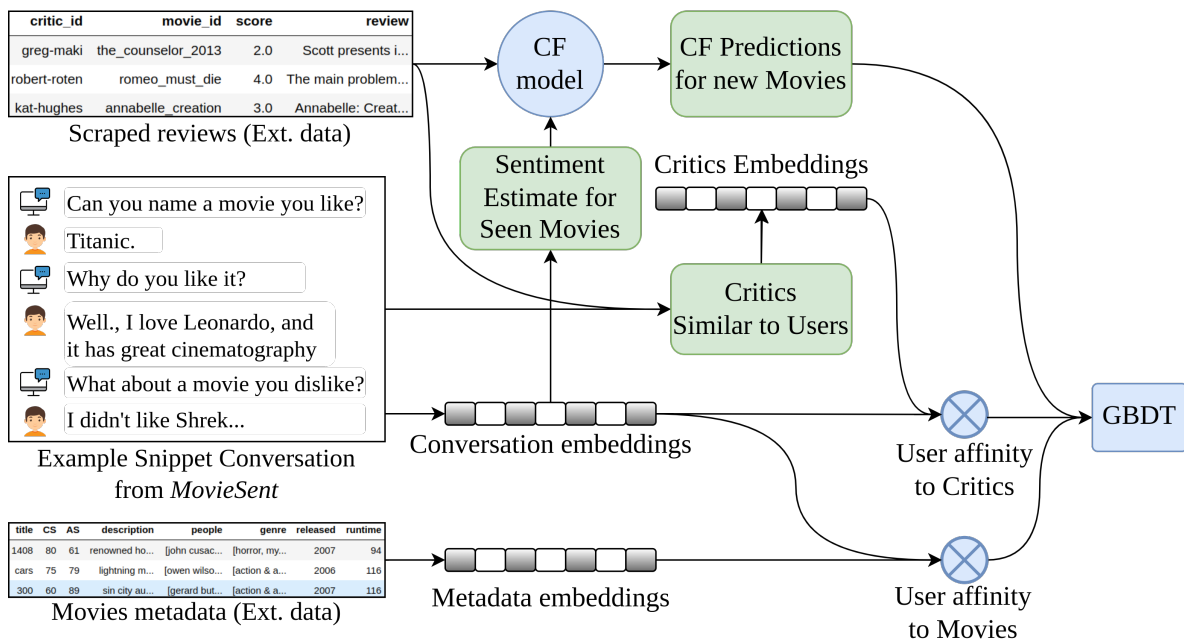


Figure 3.1: Overview of the ConvExtr system for conversational elicitation and prediction of movie preferences.

adapt the predicted baseline CF score from the critics' space to the users' space. The features are described below.

### Affinity to Critics

One of the features is the affinity between the user and critics most similar to them. For each movie, the critics are ranked based on their reviews for that movie using BM25 and normalize the returned scores. Each critic is then represented as an average of all their embedded reviews. The movie representation is a weighted average of the top most similar to user critics, where weights are the normalized BM25 scores. Finally, the cosine similarity and earth movers distance between the user representation (conversation) and movie representation (weighted average of reviews) are calculated. This results in 2 features.

### Affinity to Items

Another set of features is the affinity between the user and the movie, calculated as the semantic similarity (cosine similarity) between the user's conversation and the movie's metadata, all in embeddings form. This results in 10 additional features.

### 3.1.3 Model

Finally, all features are passed into a Gradient Boosted Decision Tree to produce the predicted score for the user-item pair.

The results can be seen in Table 3.2. Simple point averages were used as baselines.

	Conv ID	Utt number	Wizard Utterance	User Utterance	Entity	Judge Label
0	CCPE-0126d	3	Perfect! Now, what would be one of your favorite movies?	I love Mr. and Mrs. Smith. That's a great one.	Mr. & Mrs. Smith	3
1	CCPE-0ef1f	22	I can see why you wouldn't be interested in those, have you seen the shape of water	I started watching that, but I just couldn't get into it enough to finish it.	The Shape of Water	-2
2	CCPE-0d49b	9	Have you seen Bridesmaids?	Nope.	Bridesmaids	None

Figure 3.2: Example of utterances labeled with sentiment scores

## 3.2 Data

In this section, I describe the datasets used. The statistics are reported in Table 3.1

### 3.2.1 *MovieSent*: Sentiment Elicitation Dataset

The conversational Movie Sentiment Elicitation Dataset (*MovieSent*) is an extension to the dataset released in [87], which consists of Preference Elicitation conversations between “coached” crowd workers playing the roles of Wizards and Apprentices. However, the movies mentioned in the dataset were not linked to a unique identifier, which required additional manual annotation to benefit from external knowledge. Hence, all movies were manually labeled in the dataset with their RottenTomatoes IDs (RtID). Then, human annotators were asked to label each user response with a sentiment score on a  $[-3; +3]$  scale and a “None” score. The labeling was done by 8 independent judges with a 20% overlap (at most 2 people labeled the same sample). Inter-rater reliability for judges’ agreement on the labels was calculated using Cohen’s kappa [21] for binary labels, which is standard for this task, and it was 0.90 on 238 samples, indicating substantial inter-rater agreement. Reliability for the numerical sentiment was measured using a weighted Kappa [22], which was 0.77 on 163 samples. An example can be found in Figure 3.2.

### 3.2.2 Reviews dataset

Most of the movie rating datasets that existed at the time of writing the paper were not suitable for this task. Therefore, a new dataset had to be created. The basis of the CF system was critics’ ratings from an external source, specifically, a popular website, RottenTomatoes. To construct the corpus, for each movie in *MovieSent*, unique RtIDs were retrieved for critics who left reviews on that movie’s page. All the reviews that those critics have ever left for any movie were retrieved and their numerical ratings were normalized to a discrete scale from 1 to 5.

## 3.3 Metrics

Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are used to evaluate the performance of proposed model.

Table 3.1: Statistics of the datasets used

(a) Reviews dataset		(b) MovieSent dataset	
reviews	715,766	conversations	489
critics	3,664	sentiment labels	2,488
median reviews per critic	34	unique entities	712
unique movies	42,423		

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Where  $y_i$  is the true rating,  $y_i$  is the true rating for sample  $i$ ,  $\hat{y}_i$  is the predicted rating, and  $n$  is the number of samples.

## 3.4 Experiments

In this section, I first describe the baselines used for comparison and then which experiments were performed to validate the proposed method.

### 3.4.1 Baselines

First, let’s establish the natural baselines:

- *AverageCritics*: Critics’ score from RT, which is a popularity proxy in a cold start problem
- *AverageAudience*: Audience score from RT, another popularity proxy, which potentially is closer to Conversational Users than Professional Critics

These scores are readily available and require no additional implementation or training. As a widely understood metric, average rating provides a common ground for comparing the performance of more sophisticated algorithms. It is particularly helpful in the context of Cold Start when the user has no historical ratings or reviews, which is exactly the case.

### 3.4.2 ConvExtr

I now describe the experiments to validate the proposed method for inferring user preference for unseen movies from conversations to address RQ1.

The first experiments performed involved collaborative filtering algorithms used to predict the user’s rating for the unseen movie. The performances of KNN, SVD, and SVD++ were compared. The metrics for different CF models are listed in Table 3.2.

BM25 is employed to retrieve the most similar critics. It is a basic but very powerful technique. The similarity between the user and those critics is calculated by using the cosine similarity between the user representation (conversation embeddings) and

critics representations (embeddings of their reviews). I have experimented with different options for representing the user; the final version included using both user and system utterances and embedding keywords extracted from those utterances.

### 3.4.3 Evaluation Procedure

To conduct an informative evaluation of proposed methods, conversations in MovieSent are restricted to include only those with at least 3 movies with IDs mentioned in separate utterances, each with reviews in the corpus described above. The resulting conversational dataset contained 238 conversations out of the initial 489. All experiments were conducted using 5-fold cross-validation, with 48 conversations on average in each split.

## 3.5 Results

The results are shown in Table 3.2.

AverageCritics baseline performed much worse than AverageAudience, which supports the initial hypothesis that professional paid critics differ from regular users.

The best pure CF score was achieved using SVD++ with an RMSE of 1.14 and MAE of 0.92. When adding new features (the affinities), the result improves significantly, which also supports the initial hypothesis. The best score for proposed method was achieved using GBRT with an RMSE of 1.09 and MAE of 0.84.

The best sentiment model estimates user sentiment towards the movies mentioned in a conversation with an RMSE of 0.88 on a 7-score Likert scale.

The “best possible” score was calculated (also shown in Table 3.2), which is the score that would be achieved if an oracle reported the real user preferences towards the mentioned movies instead of training a sentiment estimation model. It is evident, there is much room for improvement.

Possible improvements for the future work could include using more advanced models for sentiment estimation, such as LLMs, since the “Best possible” or the oracle sentiment model has such better performance. Another improvement could be to use more advanced models for CF, such as Neural Collaborative Filtering, which could potentially improve the performance of the CF model beyond the SVD++ baseline.

## 3.6 Summary

In this chapter, a novel method for estimating user preferences from conversations was proposed. The results demonstrated that incorporating conversation content to select a more similar group of users for Collaborative Filtering improves the recommendation performance, compared to using the inferred ratings alone.

Table 3.2: Main results for *ConvExtr* (best in bold), significance from AvgAudience baseline marked with “\*”

Model	RMSE	MAE
<i>Baseline methods:</i>		
AverageCritics	1.34	0.99
AverageAudience	1.24	0.95
<i>ConvExtr (my method):</i>		
KNN (raw CF)	1.20	0.94
SVD (raw CF)	1.18*	0.95
SVD++ (raw CF)	1.14*	0.92
GBRT (final version)	<b>1.09*</b>	<b>0.84</b>
<i>Best possible:</i>	0.84	0.64

### 3.7 Limitations

The quality of the training and evaluation was compromised due to the data shortage. Since the method relies on metadata and reviews, every movie mentioned should be linked to an external ID. Existing datasets of conversational recommendations, such as ReDIAL [67] or INSPIRE [41], use item IDs for knowledge bases like DBpedia or IMDB. However, these databases do not contain reviews, which are the backbone of the approach. Therefore, I had to switch to a different knowledge base, RottenTomatoes, and manually scrape the reviews together with the movies’ metadata. The dataset utilized, CCPE [87], was manually annotated with RottenTomatoes movie IDs (RtIDs), which restricted the number of conversations available in the resulting dataset. A larger dataset of conversations annotated with IDs linked to a review database should be constructed for more robust training and accurate evaluation. The dataset annotation was manual since some movies share the same names, and the context like year or director had to be looked up in the conversation manually to resolve the correct entity. Nowadays, LLMs can potentially be used to automate this process, which would significantly reduce the time and effort required to create such a dataset, and also increase the size of the dataset drastically.

## 4. Knowledge Injection for Better Recommendations

In this chapter, I describe my work to address RQ2: “How to better represent users and items using structured and unstructured knowledge to improve the quality of recommendations?”. More specifically, “How should the knowledge be injected into the Knowledge Graph?” and “Which information about the users and items is most useful for recommendations?”. The content of this chapter is based on the paper “Augmenting Graph Convolutional Networks with Textual Data for Recommendations” [109]<sup>1</sup>, published at ECIR’23. In this paper, a novel method of integrating textual information into a Graph Convolutional Network (GCN) called *TextGCN* is proposed, based on a popular framework, LightGCN [43].

Graph neural network (GNN) approaches to recommendation models have grown in popularity in recent years [124], which is natural since so much of the information in these systems is easily mapped to a graph structure. While there is still some controversy over whether graph-embedding methods outperform more conventional recommendation systems [26], the appeal of GNN systems is strong. It has long been clear that side information and additional knowledge, typically social connections between users or structured knowledge about items, enhance any recommendation system [111]. However, the use of *unstructured* information about items or users has lagged despite the availability of vast quantities of unstructured text in the form of user reviews and item descriptions. I am aware of only a couple of examples where such unstructured information has been used in GNN recommender systems [99].

The intuition is that unstructured review text and item descriptions capture a great deal of semantic and behavioral information unavailable from the purely topological structure of a user-item interaction graph. It is posited that this unstructured text may also contain information that cannot be found in conventional knowledge graphs either. For instance, particular users may express what they like about items differently. We not only want to find similar users in terms of *what* items they like or what actors or characters, or attributes they seem to gravitate towards. We want to find similar users in terms of *how* they describe those items and attributes.

At the same time, many GNN recommender systems are increasingly complex, while in at least some cases, it has been shown that the sophisticated mixing and attention mechanisms used might even hinder recommendation accuracy [44]. Therefore, we seek to take the simplest possible approach to incorporate unstructured review and item description data into a GNN framework. The experiments show that a simple means of incorporating unstructured text into a GNN recommender improves the performance of a popular baseline system, LightGCN [44], by a similar amount as much more sophisticated

---

<sup>1</sup>The code for the model is available at <https://www.github.com/sergey-volokhin/textgcn>

approaches. In summary, the contributions are:

1. ways to augment interaction-based Graph Recommender Systems with textual information for improved node representation are explored and a simple and general approach for integrating both graphical and textual representations of users and items is introduced.
2. the effectiveness of the combined model for recommendation performance is experimentally demonstrated, with 6-21% improvements across the evaluation metrics.

I also release the code<sup>2</sup> to the scientific community for transparency and reproducibility.

## 4.1 Method Overview

This section describes the proposed approach and the features used. The architecture of the model can be seen in Figure 4.1.

The blue blob represents the frozen backbone model, which is pre-trained on the user-item interaction graph.

The yellow blobs represent the textual similarity features. Each feature consists of user and item representations (green blobs, with the top ones standing for user representation and the bottom for item representation). The dot product is applied to those two representations to get a single number, which is used as a feature for the regression layer (orange blob), which produces the final score for that user-item pair, signifying the strength of the link between them.

More specifically:

- the user review representation is obtained by taking the average of the embeddings of all their reviews.
- the user description representation is obtained by taking the average of the embeddings of all the descriptions of the items that the user has reviewed.
- the item review representation is obtained by taking the average of the embeddings of all the reviews of that item.
- the item description representation is the embedding of the description of that item.

### 4.1.1 Training LightGCN

LightGCN is a popular model with existing code available, and it is also very fast to train. It is trained on the user-item interaction graph and use it as a starting point to measure how much the structural information achieves when used alone. Next, the resulting embeddings are frozen so they will not be trained when introducing the textual features. We also experimented with back-propagating the error signal through the LightGCN model (unfreezing the model), but better results were achieved with the

---

<sup>2</sup><https://github.com/sergey-volokhin/TextGCN>



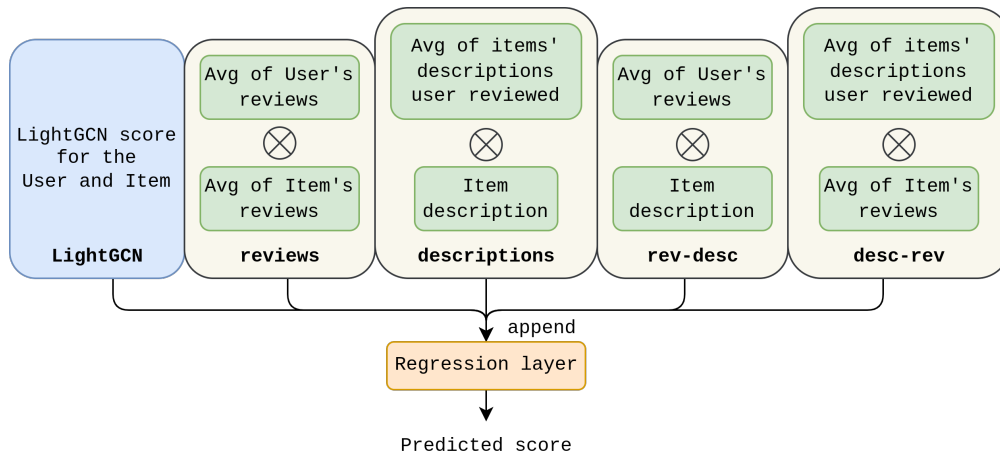


Figure 4.1: The architecture of TextGCN. Five features are used, 4 of which involve textual information (reviews and item descriptions), shown in yellow blobs. The LightGCN model is pre-trained on a user-item interaction graph and frozen, so the first feature does not change when training the regression layer, shown in the blue blob.  $\otimes$  represents the dot product.

frozen embeddings. Therefore, only experiments with frozen embeddings are described below.

The score predicted by LightGCN is used as one of the features in the model. It is shown in the blue blob in Figure 4.1.

### 4.1.2 User and Item Representation

The textual information consists of user-generated reviews for items and item descriptions. Therefore, the users and items can be represented in 2 different ways:

- represent *items* by using either the average of their descriptions or the reviews written about them.
- represent *users* by using the average of the reviews they have written or the average of the descriptions of the items they have reviewed.

Those representations are used to create features for the model. Each feature is constructed by applying dot product on different user and item representations and is then fed into a regression layer that estimates the score for that user-item pair.

The four resulting text-based features are shown in yellow blobs in Figure 4.1.

## 4.2 Data

The evaluation was done on 4 large-scale datasets of Amazon reviews [81] in the following diverse domains: Books, Electronics, Movies, and Toys. The datasets, are subsampled, taking the core-10 subset, which ensures that all items and users have at least 10 interactions. The statistics for the datasets are in Table 4.1.

Table 4.1: Data statistics (centralities are multiplied by  $10^4$ )

Domain	users	items	samples	sparsity	degree centr.		eigenvector centr.	
					mean	median	mean	median
Books	92k	58k	2.7M	99.949%	1.79	1.06	7.98	1.43
Movies	268k	78k	3.1M	99.961%	1.09	0.50	9.80	4.55
Toys	64k	32k	0.75M	99.963%	1.32	0.83	12.36	5.40
Electronics	139k	40k	2.1M	99.984%	0.43	0.23	4.45	1.62

### 4.3 Metrics

The metrics used to evaluate the proposed model’s performance are Recall, Precision, Hit Rate, and nDCG. All metrics are calculated at  $k = 20$ .

$$Recall@k(u) = \frac{|R_k(u) \cap T(u)|}{|T(u)|}$$

$$Precision@k(u) = \frac{|R_k(u) \cap T(u)|}{k}$$

$$HitRate@k(u) = \mathbb{I}(|R_k(u) \cap T(u)| > 0)$$

$$nDCG@k(u) = \frac{DCG@k(u)}{IDCG@k(u)} = \sum_{i \in R_k(u)} \frac{2^{rel_i} - 1}{\log_2(i + 1)} / \sum_{i \in R_k(u)} \frac{2^{rel_i^{ideal}} - 1}{\log_2(i + 1)}$$

Where  $R_k(u)$  is the set of top- $k$  of items recommended to user  $u$ ,  $T(u)$  is the set of items the user interacted with,  $rel_i$  is the relevancy of  $i$ -th item in the returned ranking, and  $rel_i^{ideal}$  is the relevancy of  $i$ -th item in the ideal ranking (where all the relevant items come first).

I ensure that all the items that appear in the training set get a score  $-\infty$  in the test set, so they are not recommended to the user.

## 4.4 Experiments

Extensive experiments were conducted to support the claims.

In this section, I first describe the baselines used for comparison and then which experiments were performed, adding unstructured text representations to those baselines.

### 4.4.1 Baselines

#### Collaborative Filtering Baseline

The first baseline does not use graphs and works as a sanity check to ensure that a basic Collaborative Filtering model does not outperform the proposed, more complex approach. The “implicit” [32] Python library is used to build several CF systems from the user-item interaction matrix.

Alternating Least Squares [49, 101], Bayesian Personalized Ranking [92], and Logistic Matrix Factorization [51] algorithms were tried, all of which are based on Matrix Factorization paradigm. The BPR has shown the best results in all cases. The results of it are shown in Table 4.2, marked as “CF (BPR)”.

## Graph-Based Baselines

We experiment with LightGCN [43] and several of its derivatives as baselines. LightGCN simplifies its predecessor, NGCF, by removing several unnecessary transformations and streamlining the training. Details are shown in Formula 4.1. It uses 3 graph propagation layers with a mean aggregation over neighbor nodes, normalized symmetrically by the degree of each node ( $\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$ ). The final node representation is a simple average over the three layers’ outputs.

$$\begin{aligned}
 e_u^{(k+1)} &= \sum_{i \in \mathcal{N}_u} \frac{e_i^{(k)}}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} \\
 e_u^{final} &= \frac{e_u^{(0)} + e_u^{(1)} + e_u^{(2)}}{3}
 \end{aligned}
 \tag{4.1}$$

It was chosen for its simplicity and robust results, it has shown state-of-the-art performance on several datasets, and it is also very fast to train. It has also shown better results than other graph-based models we experimented with, namely GCN [55], GATv1 [107], and GATv2 [12], and GraphSAGE [38]. All of them performed worse than LightGCN, so we decided to stick with LightGCN. The results for all the Graph Models are shown in Table 4.2.

## Single Layer

In the original LightGCN paper, the best model for the Books’14 data uses only the outputs from the final (i.e., third) layer instead of the aggregation of all. The authors called this the “Single” variation.

### 4.4.2 TextGCN

Regarding the proposed model, the following experiments were conducted:

- Compared a wide variety of graph-based models and pure CF before settling on using LightGCN as a basis for the proposed model.
- Experimented with alternate aggregators and varied the number of layers within LightGCN. The default configuration was most optimal.
- Tried several ways of combining textual features with LightGCN vectors and scores, like concatenating the user LightGCN vector with user textual representation. None of those attempts yielded any improvements.

- Experimented with back-propagating the error signal through the LightGCN model (unfreezing the model) but achieved better results with frozen graph and text embeddings.
- Introduced several non-text-related improvements that boosted the performance of both the baseline model and ours:
  - changed activation function to SELU
  - introduced Dynamic Negative Sampling (DNS)

### Dynamic Negative Sampling

The authors of the LightGCN paper have noted [43] that more advanced negative sampling techniques could improve LightGCN, and we decided to try one such sampling method to evaluate whether the improvements obtained using additional unstructured text would still appear when using improved sampling methods. Following [134], 1000 random items are ranked for each user, 40 highest ranked items are picked, 5 random positives are picked, and the training is performed on the Cartesian product of those 2 sets (200 samples per user). The intuition behind this method is that all unobserved items should have a lower score than any observed item during training since those unobserved items are either negative or, even if they are positive, they should have a lower score than the observed ones.

## 4.5 Results

The scores for different models tested can be seen in Table 4.2.

A good sanity check that the experiments make sense is that the result of GATv2 is better than that of GATv1, albeit by a small margin.

The “Single” version of the model did not perform as well as the version that takes the average of all layers, and, in fact, it performed worse than most of the other graph models.

Surprisingly, the Collaborative Filtering baseline, which did not use any graph convolutions, has shown better results than all of the graph-based models save LightGCN. This is an interesting result, that the newer, more complicated models do not necessarily transform into better results.

LightGCN beats other graph-based models by a significant margin, and variations of the proposed model TextGCN show further improvements over LightGCN.

The main results over all datasets are in Table 4.3. Each model with text outperforms its corresponding baseline without text on every metric on all the datasets. Adding Dynamic Negative Sampling further boosts the performance of both the LightGCN and TextGCN. This supports the hypothesis that the text of user reviews and descriptions contains useful information beyond what is available in the user-item graph.

Table 4.4 shows which features the models use and the average weights in the prediction layer for each feature for all the TextGCN models trained. Those weights can act as proxies for feature importance. Notice that the highest weights are for the

Table 4.2: Results for all the Baseline and Experimental models compared to the proposed model; runs over the Amazon Books dataset. Names of TextGCN models reflect different variations tested. Metrics are calculated @20 and averaged over 5 runs.

Model	Recall	Precision	Hit rate	nDCG
<i>CF (BPR)</i>	0.1422	0.0391	0.4662	0.1029
<i>Graph Baselines:</i>				
SAGE	0.0963	0.0263	0.3479	0.0674
GAT	0.1366	0.0359	0.4452	0.0984
GATv2	0.1384	0.0364	0.4503	0.0993
GCN	0.1419	0.0374	0.4584	0.1029
“Single”	0.1162	0.0318	0.4081	0.0833
LightGCN	0.1690	0.0455	0.5210	0.1244
LightGCN w DNS	0.1813	0.0490	0.5467	0.1353
<i>TextGCN:</i>				
XGBoost	0.1539	0.0372	0.4736	0.1075
GBDT	0.1749	0.0453	0.5308	0.1304
Linear	0.1833	0.0460	0.5308	0.1350
Linear w DNS	<b>0.1923</b>	<b>0.0485</b>	<b>0.5481</b>	<b>0.1428</b>

Table 4.3: Main results for all 4 tested datasets. All metrics @20. All results are statistically significant ( $p \ll 0.001$ ).

Model	Books				Toys			
	recall	precis	hit	nDCG	recall	precis	hit	nDCG
LightGCN	0.1700	0.0429	0.5044	0.1222	0.0988	0.0107	0.1787	0.0571
TextGCN	0.1833	0.0460	0.5308	0.1350	0.1160	0.0125	0.2064	0.0693
LightGCN w DNS	0.1822	0.0463	0.5290	0.1330	0.1114	0.0122	0.2035	0.0662
TextGCN w DNS	0.1923	0.0485	0.5481	0.1428	0.1268	0.0136	0.2258	0.0762
Model	Movies				Electronics			
	recall	precis	hit	nDCG	recall	precis	hit	nDCG
LightGCN	0.1575	0.0195	0.3074	0.0939	0.0543	0.0058	0.1087	0.0299
TextGCN	0.1723	0.0212	0.3321	0.1107	0.0640	0.0067	0.1261	0.0392
LightGCN w DNS	0.1789	0.0227	0.3475	0.1123	0.0703	0.0078	0.1432	0.0474
TextGCN w DNS	0.1895	0.0239	0.3644	0.1241	0.0750	0.0082	0.1513	0.0514

Table 4.4: All features used in the TextGCN model and the average weights of the corresponding neurons in the regression layers across all datasets.

feature	weight
LightGCN	1.47±0.08
$U_{rev} \cdot I_{rev}$	8.66±4.45
$U_{desc} \cdot I_{desc}$	24.44±2.27
$U_{rev} \cdot I_{desc}$	-12.80±2.79
$U_{desc} \cdot I_{rev}$	-7.23±4.66

“description” feature, which calculates the similarity between the user’s average item description vector and the candidate item vector. We speculate that this is because users are attracted to similar aspects across products. Such aspects (say “lightweight”) might be implicit in a user-item interaction graph if enough users interacted with the *same* products that shared that feature. However, using text descriptions to represent users and items appears more effective and direct. Two other textual features, using different textual representations from each user and item (*rev-desc* and *desc-rev*), have negative importance, which suggests that there is no direct useful link between the description given to the product by the seller and the reviews written by the users.

## 4.6 Summary

In this chapter, a new way to incorporate the textual features into a GCN-based recommender system was introduced.

It was shown that the text of user reviews or descriptions contains useful information beyond what is available in the user-item interaction graph. It was also shown that lightweight graph embeddings can be efficiently augmented with user-generated texts like reviews and substantially improve recommendation performance without complex models to combine the graph and textual representations.

## 4.7 Limitations

The model is trained on the user-item interaction graph, so it needs to be retrained every time new users or items are added, which is not feasible with the large number of users and items. This is a problem with many graph-based models.

Since the proposed method freezes the underlying backbone, it does not matter which particular model it is. This agnosticism means that this limitation will be solved if a backbone model that does not have that limitation is used. One possible solution is to use incremental training, which is a technique that updates the model with new data without retraining the entire model (e.g., [24, 126, 127]).

## 5. Enhancing Recommender Systems with LLM-Driven User and Item Insights

The content of this chapter is based on the work done in the last year, will be submitted as a long paper to ACL Rolling Reviews August 2024 cycle. In it I address RQ2c: “Can LLMs be used to improve user and item representations?”

A general method for data augmentation using LLMs is introduced for recommendations, which can be easily applied to any recommendation algorithm and requires a minimal amount of new parameters, similar to the TextGCN model described in §4.

To resolve ambiguity in names, I will call the:

- **TextGCN+** – a general *method* which uses textual features with a recommendation model.
- **TextLightGCN** – *LightGCN [43] model* augmented with textual features described in §4.
- **TextMMSSL** – *MMSSL [120] model* augmented with textual features and described in this chapter.

Textual data can often be noisy, incomplete, or biased. For example, user reviews can be short, poorly written, or contain irrelevant information. That makes it harder for the recommendation model to learn useful signals from the data. I hypothesize that LLMs can generate more accurate, informative, and detailed textual data, improving the user and item representations and providing a stronger signal. More specifically, I propose to generate 3 textual features to better represent the items and 1 textual feature to better represent the users, all shown in Table 5.1. The intuition for generating each particular feature is as follows:

- *Item descriptions* provided by the sellers are often biased, focusing on positive aspects and potentially omitting negative features. I hypothesize that descriptions generated with LLMs might be more objective, detailed, fluent, and accurate, hopefully resulting in improved recommendations.
- *Expert opinions* are often more analytical and critical. They can provide a different perspective on the item, which is not captured by the regular user’s reviews or the seller-provided description.
- *User archetypes* can provide a more detailed understanding of the user from the item perspective, joining the item information into a meta-summary of reviews.
- *User profiles* can provide a more detailed understanding of the user from the user perspective, joining the user information into a meta-summary of reviews.

This approach differs from the recent related work in several ways. First, I do not restrict the user profile to any specific topics or features and use the raw text generated

Table 5.1: Proposed textual features generated using LLMs

Feature	Description	Example
<i>Item Features</i>		
Item Description	General overview of the item’s features	This 27-inch 4K OLED screen has a 144Hz refresh rate
User Archetypes	Who would most benefit from the item	This laptop is ideal for graphic designers due to its high-performance GPU
Expert Opinion	What an expert in the field might say about the item	The new iPad Pro is the best tablet on the market
<i>User Features</i>		
User Profile	Which facets and aspects the user cares about most	The user prefers educational, high-quality, and interactive products

by the LLM as the user profile, unlike [69, 121, 118], which ask the LLM to infer specific user attributes like demographics, location, tags or movie styles, or fuse different user representation vectors into one.

On the other hand, we condition the generated item features to be of specific categories, namely description, use cases, and expert opinion. The hypothesis is that they would be more informative and useful for the recommendation task than the general product summaries like in [69]. Furthermore, to extract more information, user reviews about the item are supplied to the LLM instead of the pure item features, like in [69]. Finally, I show that this approach works even when most people have only one review to represent them, which is a common scenario in real life, versus other approaches, which usually require a longer history of interactions (e.g., 5 reviews for [141] and at 25 interactions for [132]).

The contributions of this work are as follows:

- Generalize the TextGCN model to be applicable to any recommendation algorithm and textual features
- Propose and validate a general method for data augmentation using LLMs for recommendations
- Show that the generated texts can be used to improve the performance of the recommender system, and that this approach is generalizable to different backbone recommendation models

## Model Notation

The features in the top TextGCN+ layer represent the models and are used as names. Each feature consists of two terms: how the user and the item are represented. Table 5.2 shows the feature notations. We generate the name for a particular model by concatenating together the beginning letters of each representation with a ‘-’ delimiter, and concatenating the features with a space delimiter. For example, the original TextGCN model that had 2 symmetric and 2 asymmetric features involving reviews and seller description is represented as: “r-r r-s s-r s-s”, and if the model only had one feature,



Table 5.2: Feature Notations. Each term in the models’ names specifies how the user or item is represented.

Category	Features
Original Features	<b>r</b> user/item reviews
	<b>s</b> seller provided original item description
Generated Item Features	<b>d</b> item description generated by LLM
	<b>e</b> expert opinion generated by LLM
Generated User Features	<b>u</b> user archetypes generated by LLM
	<b>p</b> user profile generated by LLM

where both users and items were represented by their reviews, the name would be “**r-r**”.

Similarly, I will represent TextMMSSL models as combination of features. It will consist of 2 parts: first one represents features the model was pretrained on, joined with a “+”. The second part is the features that were added in the TextGCN+ layer. The parts are concatenated with a separator (“|”). For example, the model that was *pretrained on reviews and profiles* and then *fine-tuned on the same features that original TextGCN was*, would be represented as “**r+p|r-r r-s s-r s-s**”.

## 5.1 Methodology

This section describes the methods used, implementation details, and provides links to the code and the generated texts for reproducibility and transparency.

First, I created prompts for the LLM to generate the texts. Each feature has its own prompt. Each prompt has a place for the required information to be inserted, such as the item’s description or user reviews. Context length restrictions affected the number of reviews I could use; however, since most users had only one review, it wasn’t a big problem.

I used the same framework to train the TextGCN+ model as in §4, shown in a more generalized version in Figure 5.1, without a predetermined number of features or their types. The leftmost blob corresponds to the score of the pretrained backbone model, which used to be LightGCN. The textual features remain the same as in the original TextGCN model, and consist of 2 parts each, one for the user representation vector, and the other for the item representation vector. The feature values is calculated by taking the dot product of the user and item representation vectors, shown with the  $\otimes$  symbol. Similar to TextGCN, the backbone score is concatenated together with all the textual features, and a linear layer is trained on top of that to produce the final score for the user-item pair.

The procedure is the same as for the TextLightGCN and TextMMSSL models: I train the backbone model and then add a linear layer, combining the score produced by the model and the textual features.

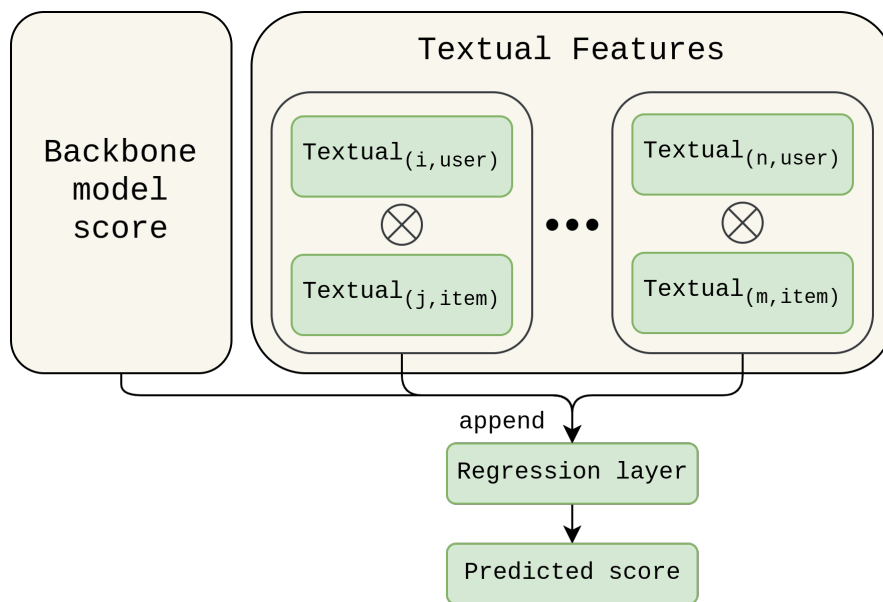


Figure 5.1: TextGCN+ generalized framework. Example of a model “i-j...n-m”

### 5.1.1 Model Training

I have tried 3 different regimes for training the TextGCN+ versions of the models:

- pretrain the backbone model, freeze it, and only train the linear layer
- pretrain the backbone model and continue to jointly train the linear layer and the backbone model
- jointly training the backbone and layer from scratch

The second regime performed best for TextLightGCN, even though the first was the best for the original TextGCN model described in §4.

For TextMMSSL, the first regime was the best, with a much smaller number of epochs required to train the model than TextLightGCN since only the linear layer was trained.

I have also experimented with using a different number of reviews to represent the users and items for TextLightGCN model. The 2 setups tried were using the median number of reviews (which was 1 for both datasets) or  $\min(x, 5)$ , where  $x$  is the number of reviews for that particular user/item.

### 5.1.2 Implementation Details

#### Texts Generation

To generate the texts I supply an LLM with the prompt and required information. For the users it is their reviews, and for the items it is their reviews together with their title and description.

When generating user profiles, I have restricted reviews to 200 tokens each to fit into the prompt, and the number of reviews used was as many as would fit into the prompt, but no more than 30.

When generating item features, I have restricted item descriptions to 1024 tokens, reviews to 512 tokens, and number of reviews was not restricted, so I have used as many as would fit into the prompt.

The texts were generated by feeding the prompts into an LLM. I started with the Llama-2 7B chat model; however, the next iteration of the Llama model was released, and I started using Llama 3 8B Instruct <sup>1</sup>. The increased context window and semantic understanding of the model significantly improved the quality of the generation. The model was loaded in bfloat16 and used FlashAttention 2 for inference, the temperature was set to 0.4, repetition penalty to 1.2, and max new tokens to 512. Generation was run on a single H100 GPU.

## Model Training

Experiments on TextLightGCN were run on 5 different seeds for both data partition and model initialization, and the results were averaged.

The code for training the TextMMSSL model was adapted from the original code for MMSSL model <sup>2</sup>. The adapted code is available on GitHub<sup>3</sup>. The embedding size was increased to 64 to be comparable with TextLightGCN results, and a learning rate of 0.001 was found with a hyperparameter search over the original dataset of Amazon Baby reviews. I used an Adam optimizer (attempting to use more advanced AdamW yielded worse performance). I used a ReduceLROnPlateau torch scheduler with patience 2, factor 0.1, and evaluated on validation recall@20 every epoch. The model was trained until the early stopping occurred, with a patience of 7 epochs. Batch size was 512, and the model was trained on a single H100 GPU. Due to the computational cost, experiments on MMSSL were run on one seed.

### 5.1.3 Alternative Backbones

I have also experimented with using other backbones for the TextGCN+ models, such as FIRE [126], GFormer [63], RGCL [97], and UniSRec [48], however, due to various reasons, results were not produced.

The authors of FIRE have provided a repository with their code, however, instead of being incremental in nature, it uses SparseSVD model, which is neither fast nor graph-based.

While the authors of GFormers provided the code for the model, it would return NaN values at arbitrary times during training with our data. I was unable to resolve the issue.

RGCL is a scoring model, not ranking, so it only trains and predicts the items with which the user has interacted. This significantly reduces the relevant data pool, and the model is not able to learn from the data. Additionally, LightGCN was not designed for this task, so I could not use it as a backbone or compare the results. Adapting the TextGCN+ framework to the scoring task might be an interesting future work.

---

<sup>1</sup><https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

<sup>2</sup><https://github.com/HKUDS/MMSSL>

<sup>3</sup><https://github.com/sergey-volokhin/TextMMSSL>

Table 5.3: Statistics of the datasets used

Number of	Toys	Movies
train interactions	263k	313k
test interactions	81k	106k
items	96k	98k
users	98k	50k
median tokens per review	26	29
reviews w < 200 tokens	97%	90%

UniSRec, a sequential recommender, has a pre-trained model released by the authors. However, it was trained in the context of descriptions and only encoded with BERT. To adapt it to our context of using reviews instead of only item descriptions and encoding text with more advanced sentiment models, I would need to either retrain the model from scratch on several domains, which was not feasible due to time constraints, or use the non-pretrained model, which has shown unsatisfactory performance.

## 5.2 Data

### 5.2.1 Source

The training and evaluation were performed on a well-established benchmark dataset, the Amazon Reviews dataset [81], with metadata associated with the items. I have chosen the Toys and Movies domains for the experiments. The metadata used consisted of the item title and description.

Instead of a commonly used approach of using a core-n sample of the data, I have decided to preserve the distribution of users with different numbers of interactions to be closer to a real-life scenario. Usually, the data has a very long tail, meaning most users only have a handful of interactions with the system. Hence, it is important to provide recommendations even for users with only one or two interactions, which is the majority of the users, and not only concentrate on the users with many interactions, who probably already enjoy the recommendations.

To do that, all users are split into 3 bins by the number of interactions: [2 – 3], [4 – 5], [6+] (users with only one interaction can not be used since at least one interaction in the training set, and at least one in the test set are required). The final statistics of the datasets are shown in Table 5.3, and the proportions of users in each bucket split are in Figure 5.2. I release the scripts to generate the specific splits I have used for reproducibility on GitHub<sup>4</sup>.

### 5.2.2 Preprocessing

The data was processed by decoding Unicode characters, unescaping and removing all HTML tags, normalizing the text using NFKD, removing all non-printable characters,

---

<sup>4</sup><https://github.com/sergey-volokhin/amazon-reviews-generated-texts>

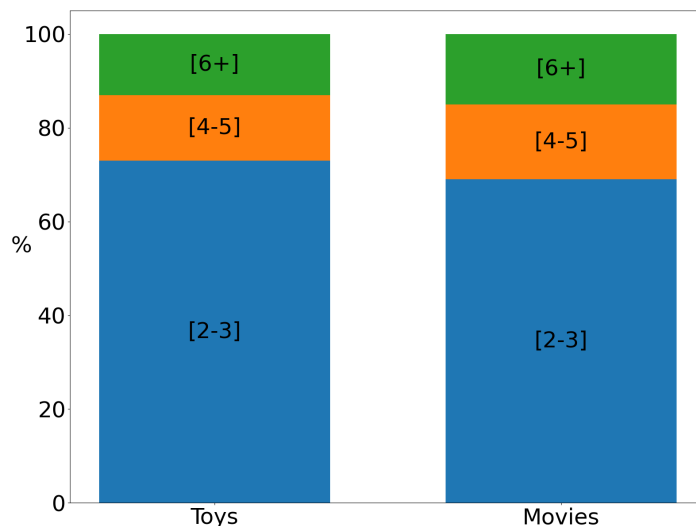


Figure 5.2: Distribution of users per bucket

and replacing multiple whitespaces with one space. After that, I removed all reviews shorter than 5 characters and those users with less than 2 reviews. Then, I calculated the proportions of users in each bucket mentioned earlier and downsampled the dataset to approx. 100k users while preserving those proportions. The final dataset was split into 75% training and 25% testing sets, split horizontally, ensuring at the same time that at least one review from each user would end up both in the training and the testing sets. Experiments for TextLightGCN were performed for 5 different train-test splits. The different data splits were generated by using shuffle split: randomly sampling interactions for each user for each seed.

## 5.3 Experiments

This section describes the experiments performed to generate the texts using LLMs and train the models.

### 5.3.1 Feature Generation

I have experimented with different prompts to generate the texts, including manually created prompts and prompts generated with GPT-4. Using the best industry practices, I have used system and user prompts and gave the model a *role* it is supposed to fulfill: **a product recommendation assistant**, with a more detailed description of the role depending on the task. The full prompts are provided in Appendix A. The list of the generated texts is shown in Table 5.1, and corresponding notations in Table 5.2.

Since the interactions in the training set differ between the train-test splits, each user in each split required a separate generation, with only those reviews in the prompt that appear in that split’s training set. However, I reduced the number of generations by caching the prompts used to generate texts for previous splits. Since most users

Table 5.4: Results of the experiments with features on **TextLightGCN**. All metrics @20. Best in bold, second best underlined.

	Toys				Movies			
	recall	prec	ndcg	hit	recall	prec	ndcg	hit
LightGCN	0.0536	0.0029	0.0364	0.0572	0.2002	0.0111	0.1469	0.2157
r-r r-s s-r s-s <sup>a</sup>	0.0593	0.0032	0.0398	0.0635	0.2034	0.0113	0.1497	0.2193
r-r	0.0592	0.0032	0.0400	0.0632	0.2025	0.0113	0.1516	0.2183
<i>Item features</i>								
r-r r-d d-r d-d	0.0589	0.0032	0.0396	0.0630	0.2046	0.0114	0.1502	0.2207
r-r r-e e-r e-e	0.0589	0.0032	0.0396	0.0631	0.2045	0.0114	0.1503	0.2206
r-r r-u u-r u-u	0.0588	0.0032	0.0396	0.0629	0.2045	0.0114	0.1501	0.2206
r-r d-d u-u e-e	0.0563	0.0031	0.0376	0.0604	0.1991	0.0110	0.1444	0.2147
all combinations <sup>b</sup>	0.0590	0.0032	0.0393	0.0631	0.1956	0.0108	0.1431	0.2108
<i>User features</i>								
p-p	0.0597	0.0032	0.0402	0.0638	0.2044	0.0114	0.1528	0.2204
r-r p-p	<u>0.0621</u>	<b>0.0034</b>	0.0415	<u>0.0664</u>	0.2053	0.0114	0.1535	0.2214
r-r r-p p-r p-p	<b>0.0628</b>	<b>0.0034</b>	<b>0.0421</b>	<b>0.0671</b>	<b>0.2079</b>	<b>0.0116</b>	0.1545	<b>0.2243</b>
<i>Combination of user and item features</i>								
p-p p-u u-p u-u	0.0578	0.0031	0.0392	0.0618	0.2032	0.0113	0.1526	0.2192
r-r p-u	0.0581	0.0032	0.0393	0.0621	0.2031	0.0113	0.1516	0.2190
r-r r-p p-r p-p p-u	0.0611	<u>0.0033</u>	0.0409	0.0652	0.2062	<u>0.0115</u>	<b>0.1557</b>	0.2224
r-r r-p p-r p-p u-u d-d e-e	0.0617	<u>0.0033</u>	<u>0.0416</u>	0.0660	<u>0.2067</u>	<u>0.0115</u>	<u>0.1555</u>	<u>0.2230</u>

<sup>a</sup> original TextGCN model

<sup>b</sup> all pairwise combinations of item features and reviews:  $[d, e, r, u] \times [d, e, r, u]$

had only two interactions, there is a 50% chance to keep the same prompt from split to split. I have saved  $\sim 66\%$  of compute that way, compared to generating everything from scratch for every split. Texts were generated for all items and users in all the training sets. I release the generated texts and the code used to generate them for reproducibility, transparency, and the scientific community’s benefit.

## 5.4 Results and Analysis

I have used the original LightGCN and MMSSL as baselines and compared versions of them augmented with the proposed textual features (TextLightGCN and TextMMSSL, respectively). The results for the baselines and the best models are shown in Table 5.6. I have also used the original TextGCN as a baseline.

Tables 5.4 and 5.5 show the results for all different features of TextLightGCN and TextMMSSL models, respectively.

Table 5.5: Results of the experiments with features on MMSSL. All metrics @20. Best in bold, second best underlined.

MMSSL features	TextGCN+ features	Toys				Movies			
		recall	prec	ndcg	hit	recall	prec	ndcg	hit
LightGCN		0.0536	0.0029	0.0364	0.0572	0.2002	0.0111	0.1469	0.2157
a_	r-r r-s s-r s-s	0.0593	0.0032	0.0398	0.0635	0.2034	0.0113	0.1497	<u>0.2193</u>
b_	r-r r-p p-r p-p	0.0628	0.0034	<b>0.0421</b>	0.0671	<b>0.2079</b>	<b>0.0116</b>	<b>0.1545</b>	<b>0.2243</b>
r	-	0.0695	0.0038	0.0404	0.0744	0.2016	0.0112	0.1441	0.2168
r	r-r	0.0698	0.0038	0.0402	0.0748	0.2024	0.0112	0.1448	0.2177
<i>Item features</i>									
MMSSL									
r+s	-	0.0602	0.0033	0.0352	0.0648	0.1929	0.0107	0.1425	0.2075
r+d	-	0.0600	0.0033	0.0345	0.0645	0.1915	0.0106	0.1387	0.2060
r+e	-	0.0616	0.0034	0.0372	0.0661	0.1902	0.0105	0.1378	0.2045
r+u	-	0.0619	0.0034	0.0379	0.0663	0.1910	0.0105	0.1399	0.2056
r+d+e+u	-	0.0369	0.0020	0.0224	0.0400	0.1706	0.0093	0.1268	0.1828
TextMMSSL									
r+s	r-r r-s s-r s-s	0.0557	0.0031	0.0335	0.0602	0.1925	0.0107	0.1422	0.2073
r+d	r-r r-d d-r d-d	0.0570	0.0031	0.0334	0.0614	0.1918	0.0106	0.1384	0.2063
r+e	r-r r-e e-r e-e	0.0575	0.0031	0.0344	0.0618	0.1899	0.0105	0.1379	0.2041
r+u	r-r r-u u-r u-u	0.0615	0.0033	0.0377	0.0659	0.1898	0.0105	0.1387	0.2041
<i>User features</i>									
MMSSL									
p	-	0.0698	0.0038	0.0402	0.0748	0.1958	0.0108	0.1412	0.2107
r+p	-	0.0720	0.0039	0.0387	0.0775	0.1960	0.0109	0.1374	0.2107
TextMMSSL									
r	p-p	0.0703	0.0038	0.0412	0.0753	0.2027	0.0112	0.1449	0.2180
p	p-p	0.0708	0.0038	0.0406	0.0758	0.1966	0.0109	0.1418	0.2115
p	r-r	0.0716	0.0039	0.0410	0.0767	0.1964	0.0109	0.1417	0.2113
r	r-r r-p p-r p-p	0.0706	0.0038	<u>0.0413</u>	0.0757	<u>0.2035</u>	<u>0.0113</u>	<u>0.1460</u>	0.2190
r+p	r-r p-p	0.0730	<u>0.0040</u>	0.0392	0.0786	0.1970	0.0109	0.1386	0.2119
r+p	r-r r-p p-r p-p	<u>0.0741</u>	<u>0.0040</u>	0.0400	<u>0.0797</u>	0.1973	0.0109	0.1387	0.2122
<i>Combination of user and item features</i>									
r+p	r-r r-p p-r p-p p-u	<b>0.0745</b>	<b>0.0041</b>	0.0402	<b>0.0802</b>	0.1972	0.0109	0.1385	0.2120
r+p+u	-	0.0628	0.0034	0.0354	0.0673	0.1840	0.0101	0.1350	0.1975
r+p+u	r-r r-p p-r p-p	0.0649	0.0035	0.0365	0.0694	0.1861	0.0103	0.1369	0.1998
r+p+u	r-r r-p p-r p-p p-u	0.0659	0.0036	0.0370	0.0705	0.1851	0.0102	0.1361	0.1987

<sup>a</sup> original TextGCN model

<sup>b</sup> best model from Table 5.4

Table 5.6: Main Results for Best Models. All metrics @20.

Metric	Toys				Movies			
	recall	prec	ndcg	hit	recall	prec	ndcg	hit
LightGCN	0.0536	0.0029	0.0364	0.0572	0.2002	0.0111	0.1469	0.2157
TextLightGCN	0.0628	0.0034	0.0421	0.0671	0.2079	0.0116	0.1545	0.2243
MMSSL	0.0720	0.0039	0.0387	0.0775	0.1960	0.0109	0.1374	0.2107
TextMMSSL	0.0745	0.0041	0.0402	0.0802	0.2035	0.0113	0.1460	0.2190

### 5.4.1 User Features

This reminder to the reader that the notation used for the models is shown in Table 5.2 and explained in subsection Model Notation §5.

The best models for all backbones and datasets have used the user profile feature, which undoubtedly shows its usefulness.

#### TextLightGCN

The single generated user feature helped, and the model improved significantly. Similar to the original TextGCN, the symmetric features ( $r-r$  and  $p-p$ ) had the highest impact. However, the asymmetric features ( $r-p$  and  $p-r$ ) have also helped, as evidenced by comparing the results of the models with and without the asymmetric features ( $r-r$   $r-p$   $p-r$   $p-p$  and  $r-r$   $p-p$ ).

Even though most users have only had one interaction in the training set (according to Figure 5.2), the generated user profile feature was nonetheless beneficial. The best model had features  $r-r$   $r-p$   $p-r$   $p-p$ .

The improvement on all metrics compared to the TextGCN baseline was, on average, 5.73% for the Toys dataset and 2.51% for the Movies dataset for the best model.

#### MMSSL

For the Movies dataset, training MMSSL on reviews is more beneficial than on profiles, as evidenced by comparing the results of the MMSSL models  $r|-$  vs  $p|-$ , as well as the TextMMSSL models  $r|p-p$  vs  $p|p-p$ , and  $r|r-r$  vs  $p|r-r$ .

For the Toys dataset, training on both profiles and reviews was best, which means that the user profiles do introduce some new useful information into the model. This is evidenced by comparing the results of the MMSSL models  $r|-$  vs  $p|-$  vs  $r+p|-$ , and the TextMMSSL models  $r|r-r$   $r-p$   $p-r$   $p-p$  vs  $r+p|r-r$   $r-p$   $p-r$   $p-p$ .

### 5.4.2 Item Features

For the Movies dataset, all the generated features separately (e.g.,  $r-r$   $r-d$   $d-r$   $d-d$ ) helped to improve the performance over the original TextGCN model without the generated feature ( $r-r$   $r-s$   $s-r$   $s-s$ ). However, the quality decreased when I combined them but removed the asymmetric features (resulting in a model  $r-r$   $d-d$   $u-u$   $e-e$ ). This further supports the hypothesis that the asymmetric features are beneficial.



For the Toys dataset, surprisingly, when running the model with reviews and any single generated item feature (e.g., **r-r r-d d-r d-d**), performances were remarkably close or lower than the original TextGCN model. Moreover, using only the “reviews-reviews” feature achieved a similar albeit slightly worse performance than the original TextGCN. Similarly to the Movies dataset, performance on the model with all symmetric features (**r-r d-d u-u e-e**) was even worse.

Finally, I tried using all combinations of the generated item features and reviews, which resulted in 16 features. I hoped the model would learn the best combination of features. However, the performance was still lower than that of the original TextGCN model.

The “all combinations” row for TextLightGCN represents a cartesian product of the set of generated item features and reviews with itself ( $[d, e, r, u] \times [d, e, r, u]$ ), which resulted in 16 features. I couldn’t run “all combinations” on MMSSL due to prohibitive computational and time costs.

### 5.4.3 LightGCN vs MMSSL

Surprisingly, a more advanced method, MMSSL, did not outperform a much simpler LightGCN on the Movies dataset. Consequently, TextLightGCN has outperformed TextMMSSL on the Movies dataset too.

MMSSL performed much better on the Toys dataset, and using only reviews with MMSSL is better than the best TextLightGCN, which uses reviews *and* profiles.

It could have been due to running MMSSL only on one seed. However, the TextLightGCN result for the same seed is even better than the average of its 5 seeds.

Another potential explanation is that hyperparameters were tuned insufficiently for MMSSL due to its computational cost. I have used the same hyperparameters for Movies and Toys, which are the same as in the original MMSSL paper for Amazon Reviews in the Baby domain.

Regardless of the underlying backbone model, adding textual features improved the model’s performance, which is a promising result.

### 5.4.4 Feature Interactions

Then, I wanted to investigate the interaction between all features, not just those generated for the items. To do that, I have run the model with different combinations of all the features available for the user and item representations. For example, it makes logical sense to use the user profile feature for the users with the user archetypes feature for the items (“p-u”). However, I discovered none of the combinations improved the model performance; only the symmetric features did.

The result for TextLightGCN experiment **p-p p-u u-p u-u** shows that user archetypes generated for items do not create a synergy with user profiles generated for users. That is further evidenced by comparing TextLightGCN experiments **r-r r-p p-r p-p** and **r-r r-p p-r p-p p-u**.

Table 5.7: Average Feature Importances in the Best Models.

Domain	Model	Base score	r-r	r-p	p-r	p-p	p-u
Toys	TextLightGCN <sup>a</sup>	1.08	3.87	2.30	3.22	3.94	-
	TextMMSSL <sup>b</sup>	1.55	2.21	1.17	0.22	2.25	-0.11
Movies	TextLightGCN <sup>a</sup>	1.04	2.88	2.08	2.28	2.88	-
	TextMMSSL <sup>c</sup>	1.40	3.22	1.17	0.57	3.27	-

<sup>a</sup> r-r r-p p-r p-p

<sup>b</sup> r+p | r-r r-p p-r p-p p-u

<sup>c</sup> r+p | r-r r-p p-r p-p

### 5.4.5 Feature Importances

Similar to §4, I have used the weights from the top linear layer as proxies for the feature importances for the models. The results are shown in Table 5.7. Once again, since the base score’s amplitude is much higher than all the other features, the lower average weight for the base score is not indicative of its importance. It would be the most important feature if scaled to the same range as the others. I have run the experiments to support this hypothesis. The results were several orders of magnitude worse than when using the score, which makes sense since most of the information is in the pretrained Graph network. Text only adds some semantic information on top of that. A similar conclusion can be drawn for MMSSL.

#### TextLightGCN

The numbers confirm the result from §4.4 that the symmetric features (where user and item representations were the same) have the highest importance. However, the relatively high importance of the asymmetric features supports the hypothesis that using those representations is still useful.

#### TextMMSSL

The best model for Movies is the same for TextLightGCN and TextMMSSL, so no separate line is needed for TextMMSSL in the results table.

Despite the model r+p| r-r r-p p-r p-p p-u achieving slightly better performance on the Toys dataset, I believe that it is only due to noise and that the real best model would be r+p| r-r r-p p-r p-p if evaluated on several seeds, since the feature importance for the ‘p-u’ feature is very close to 0 and also negative. Other results also support this, which show no significant change or even performance degradation when adding the user archetypes feature.

## 5.5 Summary

This section introduces a lightweight method for augmenting user and item representations with LLM-generated textual features. While not all proposed features are helpful,

the user profiles do benefit the recommendation task. I have also shown that even if the data is very sparse, and more than 50% of users only have one interaction in the training set, enough information can be extracted from the textual data to improve the recommendation quality.

## 5.6 Limitations

I have used Llama 3 to generate the texts, which has a very limited context window of 8k tokens. This can be a significant limitation since only partial information about the user/item can be considered during the generation of the texts. However, during writing this text, Llama 3.1 was released with a much improved 128k context, and LLM advances allow for even 1M+ token context window [28, 90]; hence, this limitation will be soon resolved without any specific interventions.

Another potential limitation is that this approach will work only for the domains and items the LLM is familiar with, so fine-tuning is required for more specialized domains.

Lastly, I have used Llama 3 to generate the texts, which is pretrained on a large portion of the internet. It is not known whether the corpus of reviews I am using to train and evaluate the model was in the pretraining data. To evaluate the performance completely fairly, I would need to have a model that has not seen the reviews, which is not currently possible. Another possible solution would be to only use newer reviews for training and evaluation, those written after the model’s training has been completed. Finally, there are ongoing efforts to make the models forget or “unlearn” [30, 129] data, which could be used to make the model forget the reviews it has seen during pretraining.

## 6. Using Textual Data for Explanation Construction

This chapter presents my work to address RQ3: How can we improve the explainability of conversational recommendations? More specifically, Which particular aspects of explanations can be improved with user-generated textual data (e.g., reviews)? The content of this chapter is based on the paper “Generating and Validating Contextually Relevant Justifications for Conversational Recommendation” [108], published at SIGIR’22.

In addition to improving the quality of recommendations, which was the focus of RQ1 and RQ2, it is also important to provide reasonable justifications. This chapter addresses this complementary aspect of recommender systems.

Recommender systems’ users benefit from understanding why or how a system came up with its recommendations [35, 47, 98, 130]. User-generated content, such as product or movie reviews, or social media posts, helps users to express their experiences and interests. Recommender systems have successfully used that content to infer preferences and improve recommendations. Such user-generated content could also help recommender systems generate finer-grained and more reliable explanations, in turn helping users make easier, more informed decisions [136].

The CRS setting poses unique challenges for generating justifications. To be effective in a conversational setting, the justifications must be appropriate for the conversation so far. Previous approaches rely on a user history of reviews and ratings of related items to personalize the recommendation. However, this information is not generally available when conversing with a new user, and as such, a cold-start problem imposes a challenge in generating suitable justifications.

The CRS setting thus requires a justification generation method that,

- First, generates contextually-relevant justifications that can be used in *conversations*.
- Second, does not depend on user history or reviews to generate justifications. Many customers have no history of reviews or other content from which to infer preferences or to find similar users’ reviews from which justifications could be generated.

To that end, a new method, **CONJURE** (CONversational JUstifications for REcommendations) to generate contextually relevant justifications for conversational recommendations is proposed and validated. Specifically, we investigate whether the conversation itself can be used effectively to model the user, identify relevant review content from other users, and generate a justification that boosts the user’s confidence in and understanding of the recommendation. To implement **CONJURE**, we test several novel extensions to prior algorithms, by exploiting an auxiliary corpus of movie reviews to

construct the justifications from extracted pieces of those reviews. In particular, we explore different conversation representations and ranking approaches. To evaluate CONJURE, we developed a pairwise crowd task to compare justifications. The experiment results show improvements in Efficiency and Transparency metrics over the previous non-contextualized template-based methods. We also release the code and an augmented conversation corpus on Github.

## 6.1 Method Overview

A natural way to create justifications is by extracting user opinions from a rich set of written reviews [66, 136, 137]. To do that, portions of reviews containing opinions about the product or its features are extracted. Elementary Discourse Units (EDUs) [74] are used as building blocks for justifications. This approach produces complete, coherent sentences, that are easier to understand than a set of keywords or phrases produced by other methods [122].

Proposed method consists of 4 basic steps:

1. Extract and preprocess EDUs
  - Extract and encode EDUs from historical reviews associated with the recommended item
  - Classify whether each EDU contains an opinion and is suitable to be used for justification generation
2. Calculate user representation using the conversation
3. Rank the positively classified EDUs from Step 1 against the user representation from Step 2
4. Construct candidate justifications and select the most natural-sounding.

Steps 1 and 4 can be solved using off-the-shelf tools; describe how exactly below. The experiments are concentrated on steps 2 and 3, i.e., how to represent the user given the conversation and rank EDUs against that representation.

### 6.1.1 Elementary Discourse Units (step 1)

A natural way to create justifications is by extracting user opinions from a rich set of written reviews [66, 136, 137]. To do that, portions of reviews containing opinions about the product or its features are extracted.

#### EDU Extraction

In [81], the authors used existing tools [117] to break the text into EDUs. As an example, consider this review (EDUs are denoted with braces, and **bold** text marks EDUs suitable for use in justifications):

“**{this is a timeless movie}**, {it really does age}. **{kubrick was the perfect director}**{to capture stephen king’s vision}”

We have used the same model, which worked well off the shelf.

## EDU Classification

For classifying EDUs, the model from [81] was fine-tuned to adapt it to the current setting. In addition to  $\sim 1600$  manually labeled samples from the original paper, 330 samples from movie-domain reviews were added. The resulting classifier achieved 0.95 accuracy and 0.91  $F_1$  using 5-fold cross-validation.

### 6.1.2 Generate and Select Candidates (step 4)

Step 3 returns 1 or 2 top-ranked EDUs, which are then used to generate the final justification. Several templates were generated, filled with those EDUs, and sorted them according to their perplexity as a proxy for fluency and coherency. The sentence with the lowest perplexity is the final answer.

For consistency, all the answers are prepended with the sentence “How about the movie %X%?” containing the recommendation item. The rest consists of generated justifications.

Some examples of generated templates are:

- “Here is what reviewers said about it: %EDU1% and also %EDU2%.”
- “This movie is %EDU1% and %EDU2%.”
- “It is a %EDU1%, with %EDU2%.”

Despite all efforts, the resulting sentences are not always as fluent as hoped. We hypothesize that this step can be further improved by prompting an LLM to come up with a natural-sounding justification that includes the mentioned EDUs.

## 6.2 Data

In this section, I describe the datasets used. The statistics are reported in Table 3.1

### 6.2.1 Conversation Corpus

Conversations from OpenDialKG [77] were used, filtered to include only those about movies and with at least one recommended item. I manually extracted movie names from the conversations and crawled `amazon.com` for matching movie titles in their Prime Video department. Unique ASINs for 450 conversations were identified.

### 6.2.2 Reviews Corpus

The reviews were downloaded from `amazon.com` for all movies mentioned in the conversations left after preprocessing. The resulting set contained 54600 unique reviews.

## 6.3 Metrics

A comprehensive set of metrics for justifications has been described previously [104] and [3] further explored how to measure whether a given explanation or justification meets a particular goal. Out of the seven metrics described in those works, the two most fitting the goals were chosen: *Efficiency* and *Transparency*. Efficiency is defined in [104] as “Help users make good decisions” and measures whether a justification helps a user to decide more quickly. Transparency is defined as “Explain how the system works” and measures how well the user understands how the recommendation was generated. Other metrics either do not fit the conversational movie recommendations setting or correlate with one of these two.

The metrics do not have a mathematical formula, hence are completely subjective, and are calculated using a Likert scale by the judges/annotators.

## 6.4 Evaluation

In this section, evaluation procedure is described and the baselines used for comparison are introduced.

### 6.4.1 Procedure

All justification configurations were evaluated using a pairwise crowd-sourced task. Previous attempts to distinguish between the different CONJURE options using point-wise ratings hinted at trends but failed to reveal anything statistically significant. Pairwise experiments force annotators to choose and, therefore, can reveal preferences in the presence of significant overall variation. The task includes screening questions to ensure that annotators are engaged in the task and are not bots. Since annotating all pairs of conversations would be prohibitively costly, we followed [37] to design the experiment and choose pairs for annotators to compare to minimize cost, and we estimated model parameter variance. The design optimizes the chosen pairs of conditions and their order. After filtering the results, we ended up with 347 unique pairs of setups covering 194 conversations and 528 justifications annotated by 67 judges. The average directional (i.e., preferring one over the other, regardless of preference strength) inter-rater reliability calculated using Cohen’s Kappa was 0.71 for Efficiency and 0.64 for Transparency.

### 6.4.2 Baselines

Three baselines were used in addition to all the different configurations of the model.

- *No Justification*: In this approach, only the recommendation was provided without explaining why the specific recommendation was made.
- *Static Templates*: Templates populated with movie metadata were developed in this method. The selection of which metadata to include was based on probabilities derived from a study focused on social aspects [83].

- *Contextual Templates*: This approach used the same templates as in the second method. However, instead of random selection, these templates were populated based on the degree of similarity between the ongoing conversation and each piece of metadata. The most similar metadata points were deemed to be the most likely choices.

The answers from the baseline methods are also prepended by the sentence “How about the movie %X%?”. Here are some examples of the templates used:

- “It has an average score %SCORE% out of 5 on Amazon Prime. Starring %ACTOR%.”
- “This movie’s main cast includes %ACTOR1%, %ACTOR2%, and %ACTOR3%. Here is the plot: %PLOT%.”
- “It is directed by %DIRECTOR%. I think you will enjoy it if you like %GENRE% movies.”

## 6.5 Experiments

This section describes the experiments performed and lists the variations of the approach compared.

Several ways to represent users were tried, as well as 2 different ways to rank EDUs (26 variations in total, including 3 baselines).

For step 2 (calculate user representation), 3 ways of using the user utterances were compared: extracting keywords in an attempt to focus the EDU ranking on the most salient information with the Python Keyphrase Extraction package (PKE) [11], extracting keywords using phrase-level sentiment analysis (Sentires) [137, 138], and not extracting anything and using the whole utterance (no extraction). Additionally, we tested whether to include only user utterances or both user and agent utterances; this had no measurable effect.

For step 3 (rank EDUs), given a set of utterances/keywords/aspects (we call them elements) that represent the conversation, we test whether it is better to find EDUs that best match each element individually and aggregate in the end or if it is better to find EDUs that best match all elements at once. If they are treated individually (*Separate*), EDUs are ranked for each element separately, the top EDU are chosen for each element, and return the EDUs with the highest overall similarity score. This approach can address different points and possibly return more diverse EDUs. If they are treated as one (*Concat*), all elements are concatenated into a single text and the similarity of EDUs to that text is computed. This approach benefits when not all aspects have corresponding EDUs and can potentially return more results.

We tested two metrics to measure the similarity between EDUs and elements. First, BM25 [93] is used with the Bag-of-Words approach. Second, Euclidean distance is computed between USE vectors for the conversation and the templates or EDUs. BM25 has the advantage of returning 0 similarity between texts that have no words in common, while USE always returns a non-zero score. Moreover, USE has the advantage of



Table 6.1: Statistically significant regression coefficients for feature-based analysis (higher is better)

Feature	Avg Coef	StdErr
<i>Efficiency</i>		
<i>Intercept</i> ( $\delta$ )	-0.314	0.089
USE	0.297	0.088
keywords	0.304	0.093
no aspects	0.207	0.096
fallback templates: 1	0.176	0.105
<i>Transparency</i>		
<i>Intercept</i> ( $\delta$ )	-0.377	0.090
USE	0.246	0.093
fallback templates: 1	0.305	0.103

returning an embedding vector to compare synonyms correctly and not relying on a string match and common words.

If only one EDU has a positive score, there is a “fallback” option, where one template with metadata is randomly picked and appended to the single EDU with the positive score. If no EDUs have a positive score, fallback to the “contextual templates” baseline.

## 6.6 Results

The static-template baseline universally performed worst for both Efficiency and Transparency. Unexpectedly, Contextual Templates were second worst, despite the earlier point-wise tests (not shown) suggesting better performance than some EDU-based systems. Eight EDU-based systems significantly outperformed baselines for Efficiency and three for Transparency. In both cases, the best systems use USE encoding for EDU ranking.

The feature-based results are shown in Table 6.1

Crowd-sourcing-based evaluation has shown that even in the absence of any user profile, justifications can be made in a conversational setting, which significantly improve upon template-based or generic baselines, by finding review fragments (EDUs) that better address stated user interests.

## 6.7 Summary

In this chapter, a method for generating justifications for conversational recommendations using a corpus of reviews is proposed. It was shown that justifications can be successfully generated using a conversation instead of more traditionally used historic reviews or numeric scores. An in-depth study on creating movie recommendation explanations by utilizing external film reviews was also conducted. In addition, the circumstances that contribute to the perception of higher-quality justifications were examined.

## 6.8 Limitations

Despite EDUs being extracted from reviews, the approach is also fundamentally template-based, which limits how the explanations can be formulated and often leads to unnatural or even completely broken responses, negatively affecting the perceived quality of the justification and recommendation.

Since EDUs are granular by nature, it is impossible to extract the more complicated reasons and arguments that might not fit into a single EDU. Additionally, the user can express preferences implicitly or in a unique way, which does not appear in the review corpus, which further limits the produced explanations.

The EDUs are picked verbatim from the reviews, we do not attempt to edit them or confirm their factual accuracy. We merely claim that at least one reviewer has expressed that opinion. This can lead to the inclusion of false information in the justifications if it was stated as a fact in the original review.

## 7. Summary and Discussion

This chapter lists the results and the insights gained from corresponding projects and papers, discusses the potential ethical implications of this work, and concludes with a summary of the thesis.

### 7.1 RQ1

This section addresses the first research question: “How to infer and represent user preferences during a conversation with the system?”

The main challenge is how to establish conversational user preferences solely from the conversation content in lieu of historical user-item interactions and how to use this information to improve the quality of recommendations.

I have proposed a method called *ConvExtr* that combines conversational context with external knowledge, such as movie reviews, to map conversational users to a set of similar reviewers and predict users’ ratings of unseen movies.

The results demonstrated that incorporating conversation content to select a more similar group of users for Collaborative Filtering improves the recommendation performance, compared to using the inferred ratings alone, which means that the textual data can provide additional information about the user’s preferences.

### 7.2 RQ2

This section addresses the second research question: “How to better represent users and items using structured and unstructured knowledge to improve the quality of recommendations?”

#### 7.2.1 RQ2(a,b)

The main challenge is how to incorporate textual features into Graph-based Recommender Systems to improve the quality of recommendations since the graph network can only support discrete nodes and not free-form text.

To address that, I have proposed a simple approach called *TextGCN* for using Graph Convolutional Networks together with textual features to improve the performance of the Graph Recommender Systems.

This method has shown promising results, with significant improvements over the baseline model that did not use text. This supports the hypothesis that the text of user reviews or descriptions contains useful information beyond what is available in the user-item interaction graph.

Furthermore, I have shown that lightweight graph embeddings can be efficiently augmented with user-generated texts like reviews and substantially improve recommendation performance without complex models to combine the graph and textual representations.

I have used the LightGCN as a backbone, but the approach can potentially be used with any graph-based recommender system.

### 7.2.2 RQ2(c)

The main challenge is to determine whether Large Language Models generate textual features that can improve the performance of Graph-based Recommender Systems, without supplying new information that was not previously available.

I have proposed a simple method to generate textual features using Large Language Models to improve user and item representations in Graph-based Recommender Systems. The method relies on textual data like user reviews or item descriptions. It aims to distill the most relevant information from the text and generate a more compact textual representation that improves the recommender system’s performance.

I have also extended the TextGCN method (now called TextGCN+) to use any backbone model, not just LightGCN, and take any textual features as input.

The results have shown that the generated textual features consistently improve the performance over the baselines that did not use LLM-generated text, even when the data used to generate the texts was also used in the original model (e.g., using reviews together with profiles generated from those reviews improved the quality over the model that only uses reviews). This supports the hypothesis that the LLMs can extract useful information from the text and provide a stronger signal to the model. The model was also evaluated on 2 different graph-based models, TextGCN and MMSSL. The TextGCN+ framework consistently improved the performance of both, which shows that the method is generalizable and can be used with any graph-based recommender system.

One possible expansion of this work is to use latent vectors to represent the users and items instead of generating the explicit text, similar to p-tuning. That might give the LLM more flexibility in generating a more faithful and relevant representation for the nodes.

## 7.3 RQ3

This section addresses the third research question: “How can we improve the explainability of conversational recommendations?”

I have introduced a novel method for generating explanations for recommendations based on conversational data.

The experiments have demonstrated that justifications can be made in a conversational setting, even without any user profile or historical interactions. By finding review fragments (EDUs) that better address stated user interests, we can improve upon template-based or generic baselines.

This can lead to more Effective and Transparent justifications for recommendations, improving the user experience and trust in the system.

Most of these findings have resulted in a publication. However, there are many points where the work can be improved or limitations addressed.

## 7.4 Ethical Considerations

Recommender systems pose several ethical challenges. Issues ranging from data privacy and security, inherent system biases, lack of transparency, digital divide, and consumer autonomy need careful examination. This section explains why the proposed work does not exacerbate existing ethical problems and refrains from introducing new ones.

### 7.4.1 Bias and Fairness

#### **ConvExtr**

The proposed method generates recommendations based on the similarity between the user and the reviewers, so the models' biases are limited to those in the reviews. The model does not introduce any new biases but can amplify the biases in the reviews.

#### **TextGCN+**

Since Foundational Models are trained on the data from the internet, they are subject to the same internal biases that humans exhibit [7, 10, 13]. Currently, attempts to mitigate those biases are ongoing. However, any work that uses those models has the potential to introduce those biases unless safeguards against that are set in place. *TextGCN+* uses such a generative Foundational Model to generate the textual features about users and items. If the model is biased, it can introduce those biases into the recommendations. However, none of the generated features are directly accessible to the user, limiting the potential harm.

#### **CONJURE**

The proposed method generates justifications for recommendations based on conversational data. While I am unaware of the original algorithm's internal biases, since the justifications are generated based on the reviews, the biases present in the reviews can be amplified.

### 7.4.2 Privacy

Traditional recommender systems, which make suggestions based on users' past behavior, inherently involve collecting and analyzing user data, leading to potential privacy issues if the data is incorrectly handled. The number of data breaches and privacy scandals is increasing every year [72, 86].

However, the proposed system does not require any historical user data to be collected. The only user information used to create the recommendations and explanations is the conversation. Those conversations are then discarded, and no user data is stored.

Another potential issue is the reviewers. On the one hand, the approaches rely on reviews that real users have written about items, and who might not have expected their reviews to be used for training machine learning models. Moreover, the datasets used are snapshots, meaning that even if the user had tried to delete their review from the Amazon website, it would still be present in the dataset. On the other hand, the datasets are publicly available and are popular in the research community, with many models trained on them throughout the years. Thus, not new privacy concerns are introduced beyond those introduced by previous models.

### 7.4.3 Transparency

Transparency is a crucial aspect of any recommender system. Users should be able to understand why a particular recommendation was made. However, many recommender systems are opaque, and the users are left in the dark about the reasoning behind the recommendations.

#### **ConvExtr**

While not directly addressing transparency, the ConvExtr model does present a potential avenue for explaining how the recommendations were made. By mapping the user to a set of similar reviewers, the system can explain that the recommendation was made based on the reviews of similar users and potentially provide examples of those reviews. This is similar to how collaborative filtering models justify their recommendations.

#### **TextGCN and TextGCN+**

Similarly, TextGCN does not directly address transparency, but the model can provide a more transparent way to explain the recommendations. The method is based on similarity calculation, so a natural way to explain a recommendation would be to provide the texts that have led to a high similarity score between the user and the recommended item, be it a description, user profile, review, or other user and item textual features, including generated.

#### **CONJURE**

While the model generates explanations for recommendations and measures Transparency as one of the metrics, the justifications are generated post-hoc, regardless of how and why the recommendation was actually chosen by the system. So the generated justifications do not directly address this issue and might be interpreted as a bad faith attempt to provide transparency. However, justifications can be useful outside of fairness auditing [139], so while we cannot claim that the justifications are transparent, they can still be useful for the user.

#### 7.4.4 Factuality in CONJURE

The justifications are generated using 2 different types of text, and it is important to separate them. One is opinionated text, extracted EDUs from the reviews, and the other is factual text, templates filled with item metadata. While the opinionated parts of the provided explanation might not be factually correct or reflect the community consensus about the item, they are nonetheless extracted from the reviews, which means that at least one real person has expressed that opinion. Hence, we do not claim that the information in the opinions used to generate the justifications is factual, merely that someone has expressed that opinion in the reviews.

### 7.5 Conclusions

In this work, I have advanced the field of Recommender Systems from different angles. I have proposed methods to improve the quality of recommendations by incorporating conversational data, external knowledge, and textual features. I have also proposed a method to generate justifications for recommendations based on conversational data. The results have shown that these methods can improve the performance of the recommender systems, including conversational ones, and provide more efficient and transparent recommendations.

To aid with the reproducibility of the results, I have released the code for all the models and experiments<sup>1</sup>. All datasets used are well-established in the field and are publicly available.

There are many promising directions to pursue based on this work. I have listed some of the limitations of all proposed models and potential ways to address them. The models can be further improved by incorporating more external knowledge, using more sophisticated language models, or combining the proposed methods into an end-to-end system. *ConvExtr* system can be adapted to use more advanced semantic similarity than BM25 and a better sentiment analysis classifier. *TextGCN* can be improved by using a more sophisticated graph neural network or, in fact, any ranking system. The *CONJURE* model can be improved by using more advanced language models to reformulate the justifications and by utilizing the vast powers of the LLMs to extract the most relevant information from the reviews.

While this work has obvious applications in e-commerce, it was evaluated on other domains as well, such as movies and books. The models were designed to be domain-agnostic, so they can be applied to any domain where user-generated or item-related texts are available, such as social media, scientific articles, or news.

---

<sup>1</sup>all available at <https://github.com/sergey-volokhin/>

# Bibliography

- [1] ANAND, Y., NUSSBAUM, Z., DUDERSTADT, B., SCHMIDT, B., AND MULYAR, A. Gpt4all: Training an assistant-style chatbot with large scale data distillation from gpt-3.5-turbo. *GitHub* (2023).
- [2] ANONYMOUS. Llm-redial: A large-scale dataset for conversational recommender systems created from user behaviors with llms.
- [3] BALOG, K., AND RADLINSKI, F. Measuring recommendation explanation quality: The conflicting goals of explanations. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (2020), pp. 329–338.
- [4] BALOG, K., RADLINSKI, F., AND ARAKELYAN, S. Transparent, scrutable and explainable user models for personalized recommendation. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval* (2019), pp. 265–274.
- [5] BALOG, K., RADLINSKI, F., AND PETROV, A. Measuring the impact of explanation bias: A study of natural language justifications for recommender systems. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (2023), pp. 1–8.
- [6] BAO, K., ZHANG, J., ZHANG, Y., WANG, W., FENG, F., AND HE, X. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems* (2023), pp. 1007–1014.
- [7] BENDER, E. M., GEBRU, T., MCMILLAN-MAJOR, A., AND SHMITCHELL, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency* (2021), pp. 610–623.
- [8] BENNETT, J., LANNING, S., ET AL. The netflix prize. In *Proceedings of KDD cup and workshop* (2007), vol. 2007, New York, p. 35.
- [9] BOGERS, T., AND KOOLEN, M. Defining and supporting narrative-driven recommendation. In *Proceedings of the eleventh ACM conference on recommender systems* (2017), pp. 238–242.
- [10] BORDIA, S., AND BOWMAN, S. R. Identifying and reducing gender bias in word-level language models. *arXiv preprint arXiv:1904.03035* (2019).



- [11] BOUDIN, F. Pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: system demonstrations* (2016), pp. 69–73.
- [12] BRODY, S., ALON, U., AND YAHAV, E. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491* (2021).
- [13] BUBECK, S., CHANDRASEKARAN, V., ELDAN, R., GEHRKE, J., HORVITZ, E., KAMAR, E., LEE, P., LEE, Y. T., LI, Y., LUNDBERG, S., ET AL. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712* (2023).
- [14] BURKE, R. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12 (2002), 331–370.
- [15] CAO, Y., MEHTA, N., YI, X., KESHAVAN, R., HELDT, L., HONG, L., CHI, E. H., AND SATHIAMOORTHY, M. Aligning large language models with recommendation knowledge. *arXiv preprint arXiv:2404.00245* (2024).
- [16] CHEN, B., GUO, W., TANG, R., XIN, X., DING, Y., HE, X., AND WANG, D. Tgcn: Tag graph convolutional network for tag-aware recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020), pp. 155–164.
- [17] CHEN, L., WU, L., ZHANG, K., HONG, R., LIAN, D., ZHANG, Z., ZHOU, J., AND WANG, M. Improving recommendation fairness via data augmentation. In *Proceedings of the ACM Web Conference 2023* (2023), pp. 1012–1020.
- [18] CHEN, X., QIN, Z., ZHANG, Y., AND XU, T. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (2016), pp. 305–314.
- [19] CHIANG, W.-L., LI, Z., LIN, Z., SHENG, Y., WU, Z., ZHANG, H., ZHENG, L., ZHUANG, S., ZHUANG, Y., GONZALEZ, J. E., ET AL. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023) (2023).
- [20] CHOWDHERY, A., NARANG, S., DEVLIN, J., BOSMA, M., MISHRA, G., ROBERTS, A., BARHAM, P., CHUNG, H. W., SUTTON, C., GEHRMANN, S., ET AL. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.
- [21] COHEN, J. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46.
- [22] COHEN, J. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* 70, 4 (1968), 213.

- [23] COSTA, F., OUYANG, S., DOLOG, P., AND LAWLOR, A. Automatic generation of natural language explanations. In *Proceedings of the 23rd international conference on intelligent user interfaces companion* (2018), pp. 1–2.
- [24] CUI, Z., SUN, X., PAN, L., LIU, S., AND XU, G. Event-based incremental recommendation via factors mixed hawkes process. *Information Sciences 639* (2023), 119007.
- [25] DAI, S., SHAO, N., ZHAO, H., YU, W., SI, Z., XU, C., SUN, Z., ZHANG, X., AND XU, J. Uncovering chatgpt’s capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems* (2023), pp. 1126–1132.
- [26] DENG, Y. Recommender Systems Based on Graph Embedding Techniques: A Review. *IEEE Access 10* (2022), 51587–51633.
- [27] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [28] DING, Y., ZHANG, L. L., ZHANG, C., XU, Y., SHANG, N., XU, J., YANG, F., AND YANG, M. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753* (2024).
- [29] EBERHARD, L., WALK, S., POSCH, L., AND HELIC, D. Evaluating narrative-driven movie recommendations on reddit. In *Proceedings of the 24th international conference on intelligent user interfaces* (2019), pp. 1–11.
- [30] EL DAN, R., AND RUSSINOVICH, M. Who’s harry potter? approximate unlearning in llms. *arXiv preprint arXiv:2310.02238* (2023).
- [31] ESTER, M., KRIEGEL, H.-P., SANDER, J., XU, X., ET AL. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (1996), vol. 96, pp. 226–231.
- [32] FREDERICKSON, B. Fast python collaborative filtering for implicit datasets.
- [33] GAO, C., LEI, W., HE, X., DE RIJKE, M., AND CHUA, T.-S. Advances and challenges in conversational recommender systems: A survey. *AI Open 2* (2021), 100–126.
- [34] GAO, Y., SHENG, T., XIANG, Y., XIONG, Y., WANG, H., AND ZHANG, J. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524* (2023).
- [35] GEDIKLI, F., JANNACH, D., AND GE, M. How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies 72*, 4 (2014), 367–382.

- [36] GOLDBERG, D., NICHOLS, D., OKI, B. M., AND TERRY, D. Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35, 12 (1992), 61–70.
- [37] GOOS, P., AND GROSSMANN, H. Optimal design of factorial paired comparison experiments in the presence of within-pair order effects. *Food quality and preference* 22, 2 (2011), 198–204.
- [38] HAMILTON, W., YING, Z., AND LESKOVEC, J. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
- [39] HARPER, F. M., AND KONSTAN, J. A. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.
- [40] HARTIGAN, J. A., AND WONG, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)* 28, 1 (1979), 100–108.
- [41] HAYATI, S. A., KANG, D., ZHU, Q., SHI, W., AND YU, Z. Inspired: Toward sociable recommendation dialog systems. *arXiv preprint arXiv:2009.14306* (2020).
- [42] HE, X., CHEN, T., KAN, M.-Y., AND CHEN, X. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (2015), pp. 1661–1670.
- [43] HE, X., DENG, K., WANG, X., LI, Y., ZHANG, Y., AND WANG, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval* (2020), pp. 639–648.
- [44] HE, X., DENG, K., WANG, X., LI, Y., ZHANG, Y., AND WANG, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval* (2020), pp. 639–648.
- [45] HE, X., LIAO, L., ZHANG, H., NIE, L., HU, X., AND CHUA, T.-S. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (2017), pp. 173–182.
- [46] HE, Z., XIE, Z., JHA, R., STECK, H., LIANG, D., FENG, Y., MAJUMDER, B. P., KALLUS, N., AND MCAULEY, J. Large language models as zero-shot conversational recommenders. *arXiv preprint arXiv:2308.10053* (2023).

- [47] HERLOCKER, J. L., KONSTAN, J. A., AND RIEDL, J. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (2000), pp. 241–250.
- [48] HOU, Y., MU, S., ZHAO, W. X., LI, Y., DING, B., AND WEN, J.-R. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2022), pp. 585–593.
- [49] HU, Y., KOREN, Y., AND VOLINSKY, C. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining* (2008), Ieee, pp. 263–272.
- [50] JANNACH, D., MANZOOR, A., CAI, W., AND CHEN, L. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.
- [51] JOHNSON, C. C., ET AL. Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems* 27, 78 (2014), 1–9.
- [52] JOHNSON, S. C. Hierarchical clustering schemes. *Psychometrika* 32, 3 (1967), 241–254.
- [53] KANG, D., BALAKRISHNAN, A., SHAH, P., CROOK, P., BOUREAU, Y.-L., AND WESTON, J. Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue. *arXiv preprint arXiv:1909.03922* (2019).
- [54] KANG, J., CONDIFF, K., CHANG, S., KONSTAN, J. A., TERVEEN, L., AND HARPER, F. M. Understanding how people use natural language to ask for recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (2017), pp. 229–237.
- [55] KIPF, T. N., AND WELLING, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [56] KOREN, Y., BELL, R., AND VOLINSKY, C. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
- [57] KOREN, Y., RENDLE, S., AND BELL, R. Advances in collaborative filtering. *Recommender systems handbook* (2021), 91–142.
- [58] KOSTRIC, I., BALOG, K., AND RADLINSKI, F. Generating usage-related questions for preference elicitation in conversational recommender systems. *ACM Transactions on Recommender Systems* 2, 2 (2024), 1–24.
- [59] ŁAJEWSKA, W., SPINA, D., TRIPPAS, J., AND BALOG, K. Explainability for transparent conversational information-seeking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2024), pp. 1040–1050.

- [60] LE, T.-H., AND LAUW, H. W. Explainable recommendation with comparative constraints on product aspects. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (2021), pp. 967–975.
- [61] LEE, D. D., AND SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature* *401*, 6755 (1999), 788–791.
- [62] LEMIRE, D., AND MACLACHLAN, A. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the 2005 SIAM International Conference on Data Mining* (2005), SIAM, pp. 471–475.
- [63] LI, C., XIA, L., REN, X., YE, Y., XU, Y., AND HUANG, C. Graph transformer for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2023), pp. 1680–1689.
- [64] LI, L., ZHANG, Y., AND CHEN, L. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems* *41*, 4 (2023), 1–26.
- [65] LI, L., ZHANG, Y., LIU, D., AND CHEN, L. Large language models for generative recommendation: A survey and visionary discussions. *arXiv preprint arXiv:2309.01157* (2023).
- [66] LI, P., WANG, Z., REN, Z., BING, L., AND LAM, W. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval* (2017), pp. 345–354.
- [67] LI, R., KAHOU, S. E., SCHULZ, H., MICHALSKI, V., CHARLIN, L., AND PAL, C. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)* (2018).
- [68] LIU, J., LIU, C., LV, R., ZHOU, K., AND ZHANG, Y. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149* (2023).
- [69] LIU, Q., CHEN, N., SAKAI, T., AND WU, X.-M. 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* (2024), pp. 452–461.
- [70] LIU, Z., WANG, H., NIU, Z.-Y., WU, H., AND CHE, W. Durecdial 2.0: A bilingual parallel corpus for conversational recommendation. *arXiv preprint arXiv:2109.08877* (2021).
- [71] LOPS, P., DE GEMMIS, M., AND SEMERARO, G. Content-based recommender systems: State of the art and trends. *Recommender systems handbook* (2011), 73–105.

- [72] MADNICK, S. Why data breaches spiked in 2023. *Harvard Business Review* (2024).
- [73] MAHMOOD, T., AND RICCI, F. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia* (2009), pp. 73–82.
- [74] MANN, W. C., AND THOMPSON, S. A. Rhetorical structure theory: Toward a functional theory of text organization. *Text* 8, 3 (1988), 243–281.
- [75] MCAULEY, J., AND LESKOVEC, J. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems* (2013), pp. 165–172.
- [76] MEI, D., HUANG, N., AND LI, X. Light graph convolutional collaborative filtering with multi-aspect information. *IEEE Access* 9 (2021), 34433–34441.
- [77] MOON, S., SHAH, P., KUMAR, A., AND SUBBA, R. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 845–854.
- [78] MUSTO, C., ROSSIello, G., DE GEMMIS, M., LOPS, P., AND SEMERARO, G. Combining text summarization and aspect-based sentiment analysis of users’ reviews to justify recommendations. In *Proceedings of the 13th ACM conference on recommender systems* (2019), pp. 383–387.
- [79] MYSORE, S., MCCALLUM, A., AND ZAMANI, H. Large language model augmented narrative driven recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems* (2023), pp. 777–783.
- [80] NAUMOV, M., MUDIGERE, D., SHI, H.-J. M., HUANG, J., SUNDARAMAN, N., PARK, J., WANG, X., GUPTA, U., WU, C.-J., AZZOLINI, A. G., ET AL. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).
- [81] NI, J., LI, J., AND MCAULEY, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (2019), pp. 188–197.
- [82] PAZZANI, M. J., AND BILLSUS, D. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*. Springer, 2007, pp. 325–341.
- [83] PECUNE, F., MURALI, S., TSAI, V., MATSUYAMA, Y., AND CASSELL, J. A model of social explanations for a conversational movie recommendation system. In *Proceedings of the 7th International Conference on Human-Agent Interaction* (2019), pp. 135–143.

- [84] PENHA, G., AND HAUFF, C. What does bert know about books, movies and music? probing bert for conversational recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems* (2020), pp. 388–397.
- [85] PENHA, G., KRIKON, E., AND MURDOCK, V. Pairwise review-based explanations for voice product search. In *Proceedings of the 2022 Conference on Human Information Interaction and Retrieval* (2022), pp. 300–304.
- [86] POOL, J., AKHLAGHPOUR, S., FATEHI, F., AND BURTON-JONES, A. A systematic analysis of failures in protecting personal health data: a scoping review. *International Journal of Information Management* 74 (2024), 102719.
- [87] RADLINSKI, F., BALOG, K., BYRNE, B., AND KRISHNAMOORTHY, K. Coached conversational preference elicitation: A case study in understanding movie preferences.
- [88] RAHDARI, B., DING, H., FAN, Z., MA, Y., CHEN, Z., DEORAS, A., AND KVETON, B. Logic-scaffolding: Personalized aspect-instructed recommendation explanation generation using llms. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining* (2024), pp. 1078–1081.
- [89] RANA, A., AND BRIDGE, D. Navigation-by-preference: a new conversational recommender with preference-based feedback. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (2020), pp. 155–165.
- [90] REID, M., SAVINOV, N., TEPLYASHIN, D., LEPIKHIN, D., LILICRAP, T., ALAYRAC, J.-B., SORICUT, R., LAZARIDOU, A., FIRAT, O., SCHRITTWIESER, J., ET AL. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530* (2024).
- [91] RENDLE, S. Factorization machines. In *2010 IEEE International conference on data mining* (2010), IEEE, pp. 995–1000.
- [92] RENDLE, S., FREUDENTHALER, C., GANTNER, Z., AND SCHMIDT-THIEME, L. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).
- [93] ROBERTSON, S. E. The probability ranking principle in ir. *Journal of documentation* (1977).
- [94] SALAKHUTDINOV, R., AND MNIH, A. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning* (2008), pp. 880–887.
- [95] SANNER, S., BALOG, K., RADLINSKI, F., WEDIN, B., AND DIXON, L. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems* (2023), pp. 890–896.

- [96] SCARSELLI, F., GORI, M., TSOI, A. C., HAGENBUCHNER, M., AND MONFARDINI, G. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.
- [97] SHUAI, J., ZHANG, K., WU, L., SUN, P., HONG, R., WANG, M., AND LI, Y. A review-aware graph contrastive learning framework for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (2022), pp. 1283–1293.
- [98] SINHA, R., AND SWEARINGEN, K. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems* (2002), pp. 830–831.
- [99] SUN, R., CAO, X., ZHAO, Y., WAN, J., ZHOU, K., ZHANG, F., WANG, Z., AND ZHENG, K. Multi-modal knowledge graphs for recommender systems. In *Proceedings of the 29th ACM international conference on information & knowledge management* (2020), pp. 1405–1414.
- [100] SUN, Y., AND ZHANG, Y. Conversational recommender system. In *The 41st international acm sigir conference on research & development in information retrieval* (2018), pp. 235–244.
- [101] TAKÁCS, G., PILÁSZY, I., AND TIKK, D. Applications of the conjugate gradient method for implicit feedback collaborative filtering. In *Proceedings of the fifth ACM conference on Recommender systems* (2011), pp. 297–300.
- [102] TAO, Z., LIU, X., XIA, Y., WANG, X., YANG, L., HUANG, X., AND CHUA, T.-S. Self-supervised learning for multimedia recommendation. *IEEE Transactions on Multimedia* (2022).
- [103] TAORI, R., GULRAJANI, I., ZHANG, T., DUBOIS, Y., LI, X., GUESTRIN, C., LIANG, P., AND HASHIMOTO, T. B. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*. <https://crfm.stanford.edu/2023/03/13/alpaca.html> 3, 6 (2023), 7.
- [104] TINTAREV, N., AND MASTHOFF, J. Explaining recommendations: Design and evaluation. *Recommender systems handbook* (2015), 353–382.
- [105] TRABELSI, F. Z., KHTIRA, A., AND EL ASRI, B. Hybrid recommendation systems: A state of art. *ENASE* (2021), 281–288.
- [106] TU, K., CUI, P., WANG, D., ZHANG, Z., ZHOU, J., QI, Y., AND ZHU, W. Conditional graph attention networks for distilling and refining knowledge graphs in recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021), pp. 1834–1843.
- [107] VELIČKOVIĆ, P., CUCURULL, G., CASANOVA, A., ROMERO, A., LIO, P., AND BENGIO, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).



- [108] VOLOKHIN, S., COLLINS, M., ROKHLENKO, O., AND AGICHTTEIN, E. Generating and validating contextually relevant justifications for conversational recommendation. In *ACM SIGIR Conference on Human Information Interaction and Retrieval* (2022), pp. 284–289.
- [109] VOLOKHIN, S., COLLINS, M. D., ROKHLENKO, O., AND AGICHTTEIN, E. Augmenting graph convolutional networks with textual data for recommendations. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part II* (2023), Springer, pp. 664–675.
- [110] VOLOKHIN, S., HO, J., ROKHLENKO, O., AND AGICHTTEIN, E. You sound like someone who watches drama movies: Towards predicting movie preferences from conversational interactions. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2021), pp. 3091–3096.
- [111] WANG, S., HU, L., WANG, Y., HE, X., SHENG, Q. Z., ORGUN, M. A., CAO, L., RICCI, F., AND YU, P. S. Graph learning based recommender systems: A review. *arXiv preprint arXiv:2105.06339* (2021).
- [112] WANG, X., HE, X., CAO, Y., LIU, M., AND CHUA, T.-S. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (2019), pp. 950–958.
- [113] WANG, X., TANG, X., ZHAO, W. X., WANG, J., AND WEN, J.-R. Rethinking the evaluation for conversational recommendation in the era of large language models. *arXiv preprint arXiv:2305.13112* (2023).
- [114] WANG, X., WEI, J., SCHUURMANS, D., LE, Q., CHI, E., NARANG, S., CHOWDHERY, A., AND ZHOU, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).
- [115] WANG, Y., CHU, Z., OUYANG, X., WANG, S., HAO, H., SHEN, Y., GU, J., XUE, S., ZHANG, J. Y., CUI, Q., ET AL. Enhancing recommender systems with large language model reasoning graphs. *arXiv preprint arXiv:2308.10835* (2023).
- [116] WANG, Y., KORDI, Y., MISHRA, S., LIU, A., SMITH, N. A., KHASHABI, D., AND HAJISHIRZI, H. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560* (2022).
- [117] WANG, Y., LI, S., AND YANG, J. Toward fast and accurate neural discourse segmentation. *arXiv preprint arXiv:1808.09147* (2018).
- [118] WANG, Z. Empowering few-shot recommender systems with large language models-enhanced representations. *IEEE Access* (2024).

- [119] WEI, J., WANG, X., SCHUURMANS, D., BOSMA, M., XIA, F., CHI, E., LE, Q. V., ZHOU, D., ET AL. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems 35* (2022), 24824–24837.
- [120] WEI, W., HUANG, C., XIA, L., AND ZHANG, C. Multi-modal self-supervised learning for recommendation. In *Proceedings of the ACM Web Conference 2023* (2023), pp. 790–800.
- [121] WEI, W., REN, X., TANG, J., WANG, Q., SU, L., CHENG, S., WANG, J., YIN, D., AND HUANG, C. Llmrec: Large language models with graph augmentation for recommendation. *arXiv preprint arXiv:2311.00423* (2023).
- [122] WU, G., LUO, K., SANNER, S., AND SOH, H. Deep language-based critiquing for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems* (2019), pp. 137–145.
- [123] WU, J., LIU, Q., HU, H., FAN, W., LIU, S., LI, Q., WU, X.-M., AND TANG, K. Leveraging large language models (llms) to empower training-free dataset condensation for content-based recommendation. *arXiv preprint arXiv:2310.09874* (2023).
- [124] WU, S., SUN, F., ZHANG, W., XIE, X., AND CUI, B. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys* 55, 5 (2022), 1–37.
- [125] WU, Y., AND ESTER, M. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the eighth ACM international conference on web search and data mining* (2015), pp. 199–208.
- [126] XIA, J., LI, D., GU, H., LIU, J., LU, T., AND GU, N. Fire: Fast incremental recommendation with graph signal processing. In *Proceedings of the ACM Web Conference 2022* (2022), pp. 2360–2369.
- [127] XIA, J., LI, D., GU, H., LU, T., ZHANG, P., AND GU, N. Incremental graph convolutional network for collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021), pp. 2170–2179.
- [128] XIAO, H. bert-as-service. <https://github.com/hanxiao/bert-as-service>, 2018.
- [129] YAO, J., CHIEN, E., DU, M., NIU, X., WANG, T., CHENG, Z., AND YUE, X. Machine unlearning of pre-trained large language models. *arXiv preprint arXiv:2402.15159* (2024).
- [130] YE, L. R., AND JOHNSON, P. E. The impact of explanation facilities on user acceptance of expert systems advice. *Mis Quarterly* (1995), 157–172.

- [131] ZAMANI, H., TRIPPAS, J. R., DALTON, J., AND RADLINSKI, F. Conversational information seeking. *arXiv preprint arXiv:2201.08808* (2022).
- [132] ZHANG, A., SHENG, L., CHEN, Y., LI, H., DENG, Y., WANG, X., AND CHUA, T.-S. On generative agents in recommendation. *arXiv preprint arXiv:2310.10108* (2023).
- [133] ZHANG, J., ZHU, Y., LIU, Q., WU, S., WANG, S., AND WANG, L. Mining latent structures for multimedia recommendation. In *Proceedings of the 29th ACM international conference on multimedia* (2021), pp. 3872–3880.
- [134] ZHANG, W., CHEN, T., WANG, J., AND YU, Y. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2013), SIGIR '13, Association for Computing Machinery, p. 785–788.
- [135] ZHANG, Y., CHEN, X., AI, Q., YANG, L., AND CROFT, W. B. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th acm international conference on information and knowledge management* (2018), pp. 177–186.
- [136] ZHANG, Y., CHEN, X., ET AL. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.
- [137] ZHANG, Y., LAI, G., ZHANG, M., ZHANG, Y., LIU, Y., AND MA, S. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (2014), pp. 83–92.
- [138] ZHANG, Y., ZHANG, H., ZHANG, M., LIU, Y., AND MA, S. Do users rate or review? boost phrase-level sentiment labeling with review-level sentiment classification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (2014), pp. 1027–1030.
- [139] ZHOU, J., AND JOACHIMS, T. How to explain and justify almost any decision: Potential pitfalls for accountability in ai decision-making. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency* (2023), pp. 12–21.
- [140] ZHOU, K., ZHOU, Y., ZHAO, W. X., WANG, X., AND WEN, J.-R. Towards topic-guided conversational recommender system. *arXiv preprint arXiv:2010.04125* (2020).
- [141] ZHU, Y., WU, L., GUO, Q., HONG, L., AND LI, J. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024* (2024), pp. 3162–3172.

## A. Prompts for Feature Generation

The following prompts, written as chat completions in JSON format, were used to generate the textual features from §5. The words surrounded by curly braces ({} ) indicate which information should be inserted.

- **User Profile:**

```
[{"role": "system",
  "content": "You are a product recommendation assistant.
  Your primary task is to identify user interests based on
  user's interaction history. It should be done in a way
  that would help the ranking system recommend more
  relevant items to users."},
{"role": "user",
  "content": "You are asked to summarize user interest based
  on the items reviewed by them. Each item is in a new
  line and contains item's title, description, user review
  , and user rating for that item.\n# Input\n### History
of items rated by user\n{history}\n\n# Task Requirement\nNow, please provide a concise and accurate summary of
the user's product preferences and the types of products
they are interested in in three sentences. Summary
should not mention any specific item names."}]
```

- **Item description:**

```
[{"role": "system",
  "content": "You are a product recommendation assistant.
  Your primary task is to generate product
  description based on the seller given description,
  item's title, and any internal knowledge you have
  about that item. It should be done in a way that
  would help the ranking system recommend more
  relevant items to users."},
{"role": "user",
  "content": "You are asked to summarize provided item
  description and its title.\n# Item Information\n<
Title>: \"{ }\"; <Description>: { }\n\n# Task
Requirement\nNow, please, provide a concise and
accurate description of the item, highlighting the
key features and benefits, focusing on the product's
specifications, functionality, and user experiences
. Aim to identify and emphasize unique selling
points (USPs) and differentiators that can assist in
accurately categorizing and matching the product
within the recommender system. Ensure the
description is clear, informative, and directly
relevant to an internal audience, encapsulating the
```

essence of the product. Limit the description to 200 words. Refrain from including the product title directly and avoid any direct communication or task acknowledgment in your presentation."}]

- **Item usecases:**

```
[{"role": "system",
  "content": "You are a product recommendation assistant.
  Your primary task is to generate potential usecases
  for the product based on its description, title,
  and any internal knowledge you have about that item.
  It should be done in a way that would help the
  ranking system recommend more relevant items to
  users."},
{"role": "user",
  "content": "You are asked to come up with potential use
  cases for the provided item based on its title and
  description.\n# Item Information\n<Title>: \"{}\"; <
  Description>: {} \n\n# Task Requirement\nNow, please
  craft a comprehensive overview that highlights item'
  s key features and practical applications in real-
  world scenarios. Illustrate its use with creative
  examples and hypothetical situations where the
  product can provide significant value or solve
  common problems. Emphasize any unique benefits and
  address potential user concerns identified from the
  reviews. Be concise, limit the description to 200
  words, presenting it in a standalone format. Refrain
  from including the product title directly and avoid
  any direct communication or task acknowledgment in
  your presentation."}]
```

- **Expert opinion:**

```
[{"role": "system",
  "content": "You are a product recommendation assistant.
  Your primary task is to generate expert opinions
  for the product based on its description, title, and
  any internal knowledge you have about that item. It
  should be done in a way that would help the ranking
  system recommend more relevant items to users."},
{"role": "user",
  "content": "You are asked to provide an expert opinion
  on the product based on the given description and
  title.\n# Item Information\n<Title>: \"{}\"; <
  Description>: {} \n\n# Task Requirement\nNow, please
  generate an expert opinion on this product. Consider
  the features, quality, market position, and any
  other pertinent aspects. Your response should
  reflect a professional assessment, highlighting
  potential strengths, weaknesses, and the overall
  value proposition of the product from an expert's
  perspectivem. Refrain from including the product
  title directly and avoid any direct communication or
  task acknowledgment in your presentation."}]
```

The user history for the user profile generation prompt was formatted using the following template, in which each item is on a separate line, and each line starts with the items number in the history:

```
n. <Item Title>: "{}"; <Item Description>: "{}"; <User Review> "{}"; <User Score>: {}
```