Graph Meets LLM for Review Personalization based on User Votes

Anonymous Author(s)

ABSTRACT

Review personalization aims at presenting the most relevant reviews of a product according to the preferences of the individual user. Existing studies of review personalization use the reviews authored by the user as a proxy for their preferences, and henceforth as a means for learning and evaluating personalization quality. In this work, we suggest using review votes rather than authorship for personalization. We suggest *MAGLLM*, an approach that leverages heterogeneous graphs for modeling the relationships among reviews, products, and users, with large language model (LLM) to enrich user representation on the graph. Our evaluation over a unique public dataset that includes user voting information indicates that the vote signal yields substantially higher personalization performance across a variety of recommendation methods and e-commerce domains. It also indicates that our graph-LLM approach outperforms comparative baselines and algorithmic alternatives. We conclude with concrete recommendations for e-commerce platforms seeking to enhance their review personalization experience.

1 INTRODUCTION

Online reviews play a central role in the success of e-commerce platforms, allowing potential buyers to gain insights about a product from customers who have already purchased it. With their growing popularity, many products accumulate a large number of reviews, making it impractical for potential buyers to traverse all of them. Review personalization aims at surfacing the most relevant reviews for each user, based on their own characteristics and preferences.

Most works on review personalization define the task as a recommendation task, where given a product and a user, the goal is to recommend the top k product reviews that would be most helpful for the user [11, 19, 20, 36]. The evaluation of such a task is nontrivial, since it requires feedback from specific users about the reviews they prefer. The common approach within review personalization literature is to leverage review authorship to gain ground truth information about user review preferences [11, 19, 43]. The underlying assumption is that reviews produced by a user are reflective of the reviews they would like to consume. The information about user-review authorship is available in many public review datasets (e.g., [35, 52]) and relying on it as a proxy for user preferences can serve as a basis for both user modeling and for creating a test set where success is defined as recommending to the user their own authored review [19].

In this work, we argue that the widespread approach of relying on authored reviews for the review personalization task has two fundamental drawbacks. First, the underlying assumption that the content produced by users is similar to the content they would prefer to consume, is questionable, as has been shown in previous work [16, 17]. It is preferable to rely on information that reflect user preferences in terms of consuming product reviews as a more direct signal. Second, review authorship provides a sparse signal. As in many other social systems, the majority of users are lurkers, i.e., they only consume reviews rather than produce them [34, 37, 42]. The entry barrier for review authorship is relatively high, compared to other types of annotations such as comments, ratings, or votes. As a result, even for non-lurkers, the number of associated reviews might be low. Recommendation approaches such as collaborative filtering can help cope with this sparsity challenge, but are still expected to perform better when the signal is more frequent and covers directly a higher portion of the users.

To overcome these two shortcomings, we suggest leveraging a different signal that associates users with reviews, reflecting their preferences for reviews they *consume*. To this end, we observe that many e-commerce platforms give users the opportunity to provide explicit feedback on reviews written by other users. This type of feedback is orthogonal to the feedback (rating) they can provide for the product itself. There are subtle differences between e-commerce platforms, where the review feedback can be a simple indication as "helpful" (Amazon, Aliexpress), a thumbs up or thumbs down for "liking" or "disliking" (Walmart), or a multi-dimensional with "helpful", "thanks", "love this", and "oh no" (Yelp). The screenshots of this functionality can be found in the Appendix. The effort required for providing a vote for a review is substantially lower than review authorship. In our analysis, we will show that not only is voting already a more widespread signal than authorship, but it also engages a considerably higher portion of the users, many of whom are lurkers, who have never engaged in writing a review. To the best of our knowledge, no previous work has comprehensively studied review personalization based on voting information.

One of the reasons likely to withhold past work from leveraging the voting signal, is merely its absence from the vast majority of popular review datasets[1] [35, 52]. To evaluate our suggested approach, we use a public dataset that does include user-review associations by voting within an e-commerce platform, over a variety of domains. We compare the use of review authorship and review voting signal for personalization across popular recommendation methods and five different e-commerce domains. Our results indicate that using the voting signal consistently yields substantially higher personalization performance. In some cases, combining the authorship and voting signals yields additional improvement over using the voting signal alone. We note that despite its exclusion from most public datasets, many leading e-commerce platforms already enable the voting functionality and naturally have access to this type of information for review personalization.

In addition to inspecting existing personalization approaches, we suggest a new personalization method, which we term *MAGLLM*. This method uses heterogeneous graph modeling to capture the relationships among products, reviews, and users. For users, large language models (LLMs) are used to enhance their representation by summarizing associated reviews. In our experiments, *MAGLLM* consistently shows higher performance results over a variety of other common recommendation techniques. Similarly to the other methods, using the voting signal is evidently preferable to using

[1] Some of the datasets include information about the total votes per review, but not the individual voters.

the authorship signal for modeling user-review relationships in *MAGLLM*. The principal contribution of this work is twofold:

- Advocating the use of voting, rather than authorship, for review personalization learning and evaluation, demonstrating its merits through data analysis, and showing it is substantially more effective over a variety of recommendation approaches
- Suggesting a review personalization approach that models product-review-user relationships using heterogeneous graph metapaths, with enhanced user representation using LLM, and showing its superiority over a variety of other recommendation methods across multiple e-commerce domains.

2 RELATED WORK

Reviews play a critical role in e-commerce by providing potential buyers with insights and influencing purchasing decisions. However, the overwhelming volume of reviews on e-commerce platforms make it difficult for users to find those that relevant to their needs. To address this challenge the platforms often use review recommendation which prioritize reviews that are not only informative but also align with a user's specific interests, filtering out noise and highlighting content that adds the most value. There are two main types of review recommendation methods - personalized, that takes the user into the consideration, and non-personalized that recommend the reviews in generic manner to all the users.

The non-personalized methods typically rank reviews based on general metrics such as relevance or helpfulness, disregarding individual user preferences or interaction history. Some of the works that deal with non-personalized reviews recommendation use machine learning techniques to automatically determine the helpfulness of reviews based on reviews textual features (e.g., [24, 40]) and sometimes even combining with products visual features (e.g., [39]). Many works focused on selecting and presenting to the user a subset of reviews covering different aspects from diverse perspectives (e.g., [47]), or consistent sentiment distribution to have a good proportion of positive and negative opinions (e.g., [25]).

In contrast, personalized recommendation models consider users' past interactions, and preferences expressed directly or indirectly through review engagement, enabling a tailored recommendation experience. Some works (e.g., [32, 33]) attempted to predict personalized helpfulness score by integrating the connections between the review author, review text, review reader, and product itself into probabilistic factorization models. However, even though the mentioned works predict the personalized helpfulness of a review, they only consider predefined user types, such as amateurs and experts, and do not consider individual users' preferences. Another line of works consider the different kinds of interactions that can be derived from reviews. For example, the relationships between the users reading the reviews and the users who wrote the reviews (e.g., [7]) or review-user-item relationships (e.g., [49]). Peddireddy [36] used recent shopping history and previous reviews as additional user information perspectives to perform reviews recommendation, however, as this kind of information is generally not available, the author constructed user profiles by randomly generating purchase histories and review engagements.

Aside from exploiting connections between users, items, and reviews, some works also incorporated the content of reviews. By analyzing the text, they were able to identify specific aspects and preferences, which were used to recommend those reviews. In a work by Suresh et al. [43], they extracted the product aspects with their sentiments from each review. A user profile is then formed based on the user's preferences for specific products and their qualities. These profiles are used, along with social networks information, to identify similar users. However, social network information is often sparse and unavailable. Dash et al. [11] grouped similar users by preferences for product aspects that were extracted using LDA and sentiment. After grouping the similar users, the review recommendation was done per group. In this case, personalization might have a smaller impact on a particular user. Huang et al. [19] also used sentiment-based recommendations. In addition, they evaluate similarity between users based on the aspects and sentiments they share. Furthermore, given an individual user, they considered reviews written by them as ground truths, and calculated similarity between those reviews and candidate reviews.

A related line of works [23, 26, 41, 44] include graph-based methods, where most works deal with item recommendations, and do not consider review recommendation. They often use user and item embeddings that were learned from the graph for rating prediction.

Most of the above mentioned works solely rely on authored reviews for the review personalization. This approach holds two main limitations, (1) a relatively small amount of users write reviews, which makes this signal sparse, and (2) the content in the authored reviews is not necessarily similar to the content the users would prefer to consume. To address the above gaps, we suggest using review votes rather than authorship for personalization. In addition, we propose *MAGLLM*, an approach that leverages heterogeneous graphs for modeling the relationships among reviews, products, and users, with LLM to enrich user representation on the graph.

3 DATASETS

In this section, we first present the primary dataset used for our analysis and evaluation. We then introduce an additional dataset, employed mainly for gaining deeper insights into user behavior, particularly how users express their preferences in reviews. Both datasets are significant because they contain valuable and non-trivial information about the feedback users provide on products and reviews, which is not commonly available in other public datasets used in previous studies.

Ciao Dataset. This is the main dataset used for the evaluation of our proposed method, and the comparison of review authorship versus review voting signal for personalization across various recommendation methods examined in our research. It is a public dataset of product rating and reviews from Ciao, a European-based online-shopping portal. Ciao uses a 6-point rating scale ranging from 0 to 5. It also allows users to express their feedback about the helpfulness of a review using numeric quality ratings of the same 6-point rating scale. We refer to this feedback hereafter as a *vote*. The dataset contains the user identifier, product name, product category, product rating score gave by the reviewer, overall review helpfulness score, written date of the review, content of the review, and the helpfulness voting score by a specific user id. We used the version of the Ciao dataset [45]. The data was crawled from the site in the month of May, 2011 and consists 27 categories (there is

Table 1: Ciao Dataset - Basic Characteristics.

#Products	#Reviews	#Reviewers	#Votes	#Voters
109,451	270,126	10,731	7,788,175	42,035

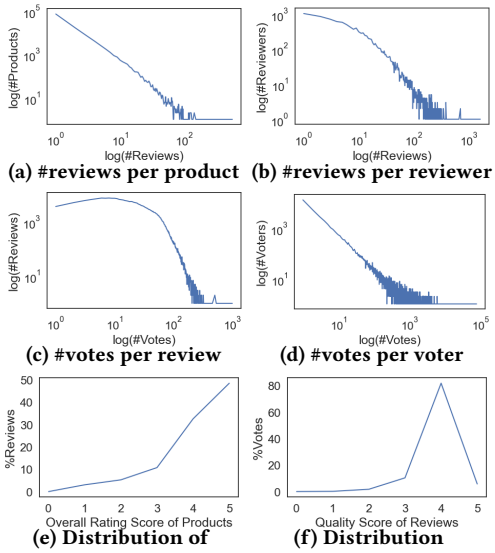
Table 2: Ciao Dataset - Review Writing and Voting.

Action	#Users	%Users
Write	10,731	24.26%
Vote	42,035	95.02%
Write & Vote	10,050	22.72%
Write & Not Vote	681	1.54%
Vote & Not Write	31,985	72.30%

a category called Ciao Cafe which is the biggest category, which was not included since the reviews in this category do not refer to real products.). Tables 1 and 2 present the statistics of the dataset. In total, there are 270,126 reviews, over 10,700 users who write reviews and over 42,000 users who vote for reviews. We can see that more than 98% of the users interact with the products by voting for reviews and only 23% of the users write reviews on products. We opted to use this dataset since it shares, for each user, the reviews they voted for (and the rating of the vote), while other datasets, such as Amazon [35] and Yelp [52] only include the total number of votes per reviews or do not contain review vote information at all, even though the corresponding platforms do feature a helpfulness score per review (e.g. TripAdvisor and IMDb [29]). As mentioned in Section 1, this information is typically available to e-commerce sites who aim at personalizing product reviews.

Figure 1 shows different distributions of reviews and votes in the dataset. In the first two figures, 1a and 1b, we can see the distributions of number of reviews per product and number of reviews per reviewer, respectively. Both distributions follow a power-law distribution. A large portion of products receives a small number of reviews while a small portion of products has many reviews. As for users, few users write a large number of reviews while many users write only few reviews. A similar trend can be seen also in Figure 1d that shows the distribution of number of votes per voter. Figure 1c shows the distribution of number of votes per review, which deviates slightly from a perfect power-law distribution since the first values, that refer to reviews with five or fewer votes, are not the most frequent in the dataset. In addition, we looked on two more distributions that refer to the reviews rating score. First, the distribution of overall rating score of products expressed in reviews (Figure 1e) that were given by the writer of the review (the reviewer) to the product. The rating score expresses how much the reviewer is satisfied with the product. Second, the distribution of the quality ratings of reviews (Figure 1f) given by the reader of the review (the voter). The quality score expresses how much the review was helpful for the reader. It can be seen for rating score that nearly 50% give the highest score of 5, additional 32.5% give 4, and only 19% give the four lower scores of 0-3. For quality, overwhelmingly the most common score is the second highest - 4 - at nearly 80%. 3 and 5 account for roughly 11% and 6% each respectively, and only 2.2% give the three lowest scores of 0-2.

Edmunds dataset. Our second dataset is also public and contains car reviews [14] collected from American Automotive online shopping site Edmunds.The total number of reviews is 42,288. Each review includes the date, the author's name, the full review text, and an additional 'favorite' field (manually filled by the author),



(a) #reviews per product (b) #reviews per reviewer (c) #votes per review (d) #votes per voter (e) Distribution of overall reviews rating (f) Distribution of quality rating

Figure 1: Ciao Dataset Analysis

which highlights aspects or characteristics to which the author paid extra attention. After removing reviews with missing data, we were left with total of 40,925 reviews. This dataset is unique because it highlights specific written reviews through the 'favorite' feature, which provides insights into the characteristics of the product that users particularly liked. Leveraging this information allows us to assess how well written reviews capture user interests in the product. To test this assumption, we later compare the full review text with the additional 'favorite' field for each user.

4 ANALYSIS

Most of the existing works in review personalization rely on modeling approaches that utilize reviews written by users themselves to create personalized review recommendations and perform evaluation [11, 19, 36, 43, 48]. Relying only on reviews written by users has two significant limitations. First, written reviews may not capture the full spectrum of user preferences and opinions. The assumption that the reviews a user writes represent, in terms of content and style, the reviews they are also going to prefer as consumers, has never been put to test. Second, as in other user-generated content, the majority of users who consume reviews do not produce ones. Table 2 presents the statistics of review writing and voting in the Ciao dataset described in Section 3. While only 24.3% of the users write reviews, over 95% of the users vote for reviews. This trend can be also observed in other review datasets that are widely used in recommendation research, such as Amazon [35] and Yelp [52].

Since the information of user-review interactions is not available in these datasets (only the total amount of votes per review), we cannot calculate directly the portion of users that write or vote. Instead, for each dataset, we calculated the total number of reviews across all products to represent the written reviews, and the total number of votes across all reviews to represent user voting. As can be seen in Table 3, in the Amazon dataset, across several popular categories voting is more prevalent than writing a review with votes exceeding reviews by more than 60%. Notably, in the CDs and Vinyl category, the ratio is nearly double. In the Yelp dataset,

Table 3: Reviews vs. Votes Statistics across Datasets.

Dataset	Category	#Reviews	#Votes	Ratio
Amazon	Books	29,475,453	52,381,607	1.78
Amazon	CDs and Vinyl	4,827,273	9,212,281	1.91
Amazon	Camera & Photo	4,340,159	7,020,382	1.62
Amazon	Exercise & Fitness	3,193,115	5,155,488	1.61
Amazon	Games	1,502,718	2,573,243	1.71
Amazon	Hair Extensions, Wigs & Accessories	985,065	1,648,990	1.67
Yelp		6,990,280	14,048,967	2.01

Table 4: Ciao 4-core Dataset Statistics across Top 5 Categories.

Category	#Products	#Reviews	#Reviewers	#Votes	#Voters
DVDs	2,640	27,964	3,703	602,291	5,189
Beauty	1,813	14,091	2,747	333,407	4,417
Food & Drink	1,318	12,084	2,533	349,302	4,410
Internet	790	11,737	3,079	226,115	4,965
Games	909	8,919	2,486	110,424	4,269

the number of votes (which includes three different types: useful, funny, and cool) is over double the number of reviews.

Even among those who write reviews, many do so rarely and therefore provide a sparse basis for personalization. Most users do not write reviews, but interact with reviews, explicitly (e.g. liking or disliking the review) or implicitly (e.g. time spent reading the review). These limitations highlight the need for more comprehensive approach that incorporate additional forms of user feedback.

To overcome the aforementioned limitations, we propose to use the user-review interactions data, specifically, like and dislike votes, that we believe are more accurately representing the user preferences and therefore can help to create a better ranking of the reviews presented to the user. To be able to estimate the effectiveness of the addition of the like signal versus the usage of solely the reviews written by the users, we use a dataset that contains the feedback of the users alongside the written reviews. For the training and evaluation of our approach we created a dataset based on the Ciao dataset, which we refer to as the "4-core" dataset. Concretely, we considered all users who wrote at least 4 reviews and all products that have at least 4 written reviews, with 4 chosen based on empirical testing. This type of dataset allows to compare two different approaches on the same set of users: *write* versus *vote*. The first approach relies only on reviews written by the user, while the second approach relies only on reviews the user has voted for. Additionally, we will test a third approach, a hybrid version of *both* write and vote, to examine if the combination of these two approaches provides any improvement over the single-signal approaches. In our work, we focus on the five biggest categories: DVDs, Beauty, Food & Drink, Internet, and Games. Table 4 presents the statistics of the categories in the 4-core dataset.

4.1 Do Written Reviews Accurately Reflect User Preferences?

To validate the assumption that written reviews do not fully represent user preferences, we analyze the cars dataset since in addition to the reviews written by users, it includes an explicit field mentioning the favorite characteristics of the reviewed car by the user. We set out to examine the intersection between the aspects mentioned in the review and the aspects mentioned by the user in the "favorite" field. We used a neural named-entity recognition tool [6] to identify a list of car aspects from the reviews text. Since the tool did not extract all tokens we considered as aspects from the text, we used in addition two public available datasets of car aspects that were manually collected and annotated from various websites with reviews [10, 30] to enhance the aspect list. We added to the list the plural form of singular aspects and vice versa to include different variants of the aspects. Then, we searched for the exact aspects tokens in the raw text. Finally, we calculated how many aspects are mentioned in both the review and the "favorite" field for each user.

The analysis shows that the average number of the aspects mentioned in the review, 8.451, is almost double the number of aspects mentioned in the "favorite" field, 4.879. Moreover, the intersection of the two list is quite low, in average 0.791, hence most of the aspects are different. Since the "favorite" field contains explicit user preferences, it shows a clear gap between the content of written reviews and the preferences of the user. Thus, written reviews do not fully represent the user preferences and relying exclusively on them may not be sufficient as some information may be missed.

5 VOTING SIGNALS FOR USER PREFERENCES

As mentioned, written reviews may not fully represent the user preferences, therefore rather than relying on the sparse and noisy *write* signal, we suggest relying on more frequent and direct vote signal. To our knowledge, we are the first to explore the use of the *vote* signal as a primary source for capturing user preferences in review recommendation. By leveraging the data from the user-review interactions, we aim to create a more accurate representation of user preferences and improve the personalized review recommendation.

To test our hypothesis end evaluate the impact of using votes as the primary signal on personalized review recommendations, we will explore three different strategies: 1) using only the *vote* signal, 2) using only the *write* signal, and 3) combining *both* signals. For each signal $\in \{vote, write, both\}$, the user is associated with a different review set according to the signal. Specifically, the reviews the user voted for in the *vote* signal, the reviews the user wrote in the *write* signal, and a combination of the two signals in the *both* signal. Accordingly, we define this set as *signal-based review set* associated with a user, and refer to this definition throughout the text. We consider reviews with scores of 3, 4, or 5 as positive or "liked" reviews, and reviews with scores of 0, 1, or 2 as negative or "disliked" reviews. When utilizing the *vote* signal, we include only positive reviews in the *signal-based review set*, as we believe they more accurately reflect the user's preferences compared to negative reviews. We will examine each of the signals using a variety of recommendation methods including our proposed approach based on heterogeneous graph model and LLM-generated user profiles.

6 OUR SUGGESTED APPROACH

We propose *MAGLLM*, a personalized review recommendation technique that models relationships among products, reviews, and users using heterogeneous graph network with meta-paths and enhances user representation using LLM. Our approach consists of two components: (1) a user profile generation using LLM to provide a rich representation of users based on the review they wrote and liked and (2) a heterogeneous graph-based model with meta-paths originally designed for the link prediction task and adjusted for the

review personalization task. The graph structure is aimed at capturing the diverse and complex relationships between users, reviews, and products. The predicted link connects a user to a review, with the score reflecting the probability that the user will like the review.

An overview of *MAGLLM* architecture is illustrated in Figure 2. First, the user's review history is given as input to the LLM, which generates a user profile summary in natural language. This profile serves as the user's representation in the graph. The profile is then converted into a vector using Word2Vec embeddings, which becomes the user node features. Similarly, the review and product nodes features are also represented by Word2Vec embeddings. A heterogeneous graph is then constructed using the users, reviews, and products as nodes, along with the connections (edges) between them and the predefined meta-paths. Finally, employing the learned embeddings of users and reviews nodes, we perform link prediction to predict the probability that a user will like a specific review.

User Profile Generation. To more accurately capture user preferences and topics of interest, we aim to generate a high-level summary of the user preferences, incorporating as many relevant details as possible. To achieve this, we leverage an LLM to generate a summary by analyzing the history of reviews a user has interacted with, whether through writing or voting. This allows the LLM to extract valuable insights into the user's preferences, such as favored products or sentiment toward specific aspects. The user profile is built independently for each category and signal by selecting relevant reviews from the *signal-based review set* linked to the user across all other categories. From this list, we randomly sample 10 reviews to create the history of the user and truncate each review to the first 150 tokens (due to LLM prompt size limitations). The profile is represented as the average of the Word2Vec embeddings of its words. We experiment with several prompt formats using different content and style, and present here the one that achieved the best performance. The LLM prompt is constructed as follows:

```
You are asked to describe user interests and preferences based on his/her
{signal} reviews list, your'e given the user's past {signal} signal reviews in
the format: <product category, product title> : <product review content>
You can only response the user interests and preferences (at most 10
sentences). Don't use lists, use summary structure. The output should begin
with the word Profile. These are the {signal} reviews:
<product-1-category, product-1-title>: <review-1-content>
…
<product-10-category, product-10-title>:<review-10-content>
```

where the {signal} token is replaced with 'liked' or 'written' or 'liked or written' according to the specific signal. We use LLaMA-7B model [46] with the implementation of llama-cpp-python [2].

Other than for user profile generation, We explored an alternative strategy for leveraging LLMs in our task by utilizing a two-stage framework consisting of retrieval and recommendation stages. This concept, introduced in the work of LlamaRec [54], was applied for sequential recommendation. In the retrieval stage, a model is used to generate an initial list of candidates, and in the recommendation stage, an LLM ranks these candidates. Similarly, in our implementation, we used a recommendation model to identify the initial candidates, and then the LLM re-ranked the retrieved candidates. Initially, we applied a listwise approach, but since LLMs have been shown to suffer from position bias [9, 18, 28, 53]—where the model gives disproportionate importance to items based on their position in the input sequence—we also experimented with a pointwise approach, in which the LLM ranked one candidate at a time. However, even with the pointwise approach, the performance remained low, and it even degraded the performance of the recommendation model used at the retrieval stage.

Table 5: Graph statistics across signals.

Signal	Nodes	Edges	Meta-paths
Vote	# Voter (V): 5,959 # Review (R): 74,795 # Product (P): 7,470	# V-R: 951,730 # R-P: 74,795 # V-P: 653,043	VRV, VRPRV, VPV, RPR, RVR, RPVPR
Write	# Author (A): 5,672 # Review (R): 74,795 # Product (P): 7,470	# A-R: 74,795 # R-P: 74,795 # A-P: 74,438	ARPRA, APA, RAR, RPR, RPAPA
Both	# Voter (V): 5,959 # Author (A): 5,672 # Review (R): 74,795 # Product (P): 7,470	# V-R: 951,730 # A-R: 74,795 # R-P: 74,795 # V-P: 653,043 # A-P: 74,438	VRV, VRPRV, VPV, ARPRA, APA, RVR, RAR, RPR, RPVPR, RPAPR

The LLM-based user profile generation showed the best performance compared to the other LLM-based strategies that we tried, thus we use the llm-based profiles as the representation of users to enrich the information in the graph.

Heterogeneous Graph. Unlike traditional graphs, where nodes and edges are of a single type, heterogeneous graph allow for the representation of multiple types of nodes (e.g., users, products, reviews) and edges (e.g., a user writing a review, or a user voting for a product). Meta-paths are a set of relationships, which represents sequences of specific node and edge types. Our problem involves diverse relationships between users, reviews, and products, which required a structure capable of modeling complex data interactions that capture rich and diverse information. A heterogeneous graph combined with meta-paths provides the necessary framework to achieve this, which is why we chose this approach for our model. Specifically, we employ MAGNN [13] implementation [1] which uses aggregation over meta-paths to incorporate information not only from the two endpoints but also from intermediate nodes along the path, and ultimately generate node embeddings. We adjusted the link prediction task for a review personalization task, where the link between a user and a specific review is used to predict the probability that the user will like the review.

For each signal $\in \{vote, write, both\}$, we constructed a heterogeneous graph with meta-paths to describe the user-review, user-product and review-product interactions. Connections (edges) between a user and a review or product were created according to the *signal-based review set* associated with a user. For instance, for signal $\in \{vote\}$, a user will be connected to reviews she voted for and the products associated with those reviews, but not to reviews she authored. Similarly to [13], we present the graph statistics of the signals in Table 5. The table shows for each signal the number of nodes and edges in the graph structure, and the meta-paths. The meta-paths were manually created based on domain knowledge and include only paths that start or end with users and reviews, as our focus is on personalizing reviews for users. The graphs consists of multiple node types including Voter (V), Author (A), Review (R), and Product (P). Using meta-paths allows the formation of complex relationships between nodes. For example, the meta-path *Voter-Review-Voter* (VRV) represent a connection between two different users who voted for the same review, and the meta-path *Author-Review-Product-Review-Author* (ARPRA) represent two users who wrote reviews for the same product. In addition, we use node content features for products, reviews, and users. Each review is represented by averaging the Word2Vec embeddings of its words. Product features are calculated by averaging the vectors of all its

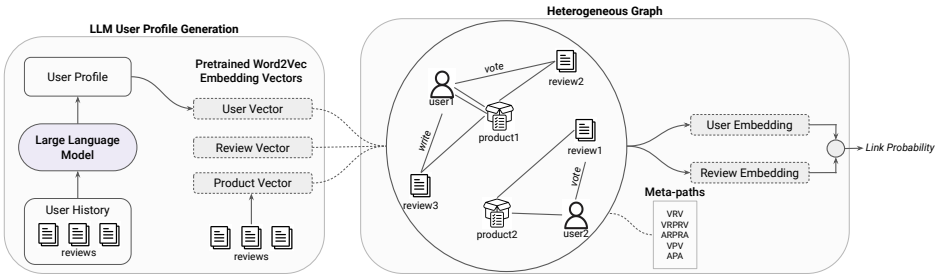


Figure 2: An overview of our proposed method *MAGLLM* for personalized review recommendation

reviews. User features are based on the LLM-generated profiles, as explained in *User Profile Generation*. We model the task of personalized review recommendation as a link prediction task, where the output of the model is the probability of a link connection between a given user and review nodes. Based on the link prediction scores, we rank the product reviews for each user.

7 METHODS

In this section, we describe the models and metrics used in our evaluation. Our goals are twofold: first, to compare the effectiveness of the voting signal (*vote*) versus the authorship signal (*write*) for personalization across various recommendation methods, from basic to more advanced ones. Second, to compare these methods with our proposed approach and show that it outperforms them all.

7.1 Models

7.1.1 *Non-personalized baselines.*

- **Random** - a method that randomly sorts the review list.
- **Popularity** - a non-personalized method that recommends most popular reviews to all users. Popularity is calculated by the number of positive votes (equal or greater than 3). While simple, popularity is known to often produce a strong baseline [21].

7.1.2 *Content-based methods.*

In this group of models, we created a user profile for each user. The profile is represented by a vector which is calculated differently in each model, as detailed below. We used Word2Vec embeddings to represent words in a review. We trained a model using CBOW with negative sampling, a vector size of 300, and window size of 8 on two review corpora, Ciao and Amazon, including multiple categories. Throughout the text, whenever we refer to word embeddings, we specifically mean the word embeddings extracted by the trained Word2Vec model.

- **Top Terms** - a method that extracts the top k frequent terms (excluding stop words) from the *signal-based review set* associated with a user to create a user profile. The profile is calculated by averaging the word embeddings of these top terms. Each review is also represented by the average word embeddings of its top k frequent terms. Finally, to rank product reviews for a user, cosine similarity was calculated between each review vector and the user profile vector, with the reviews sorted by similarity score. We experimented with different values of k and settled on $k = 20$ as it yielded the best performance.
- **Word2Vec** [31] - a model that learns vector representations of words and captures their semantic relationships. We used the model explained above and for each review, created a vector representation by averaging its word embeddings. The user profile is calculated by averaging all review vectors in the user's *signal-based review set*. Product reviews are then ranked by cosine similarity between each review vector and the user profile, with reviews sorted by similarity score.
- **Sentence-BERT** [38] - a modification of BERT [12] that finetunes it to efficiently compute sentence embeddings with a Siamese networks architecture. It generates semantically meaningful sentence vectors, making it much faster and more efficient for tasks like sentence similarity, clustering, etc. Using this model, each review is represented by the average of its sentence embedding vectors. Then, the user profile is calculated by averaging the review vectors of all reviews in the user's *signal-based review set*. We used NLTK [27] to split the review into sentences, and applied on them the pre-trained Sentence-BERT model `all-MiniLM-L6-v2`. Product reviews ranking was calculated using cosine similarity between each review vector and the user profile vector, with the reviews sorted by similarity score.
- **A2SPR** - a method based on the work by Huang et al. [19] that uses aspect sentiment similarity for personalized review recommendations. The method identifies users' aspect preferences from the reviews they wrote, calculates similarity between users with shared preferences for the same products, and ranks reviews using a helpfulness score. To implement[2] this, we extracted predefined aspect lists for each category using a semi-automated approach. Sentiment was determined via NLTK, and user similarity was modeled using a product-associated graph. The review helpfulness score then ranked the product reviews for each user. This aspect extraction process was time-intensive, since it required manual tuning to ensure the aspects were accurate, so we applied it to two categories for testing. Due to low performance, it wasn't expanded to other categories.

7.1.3 *KNN Collaborative Filtering.*

- **KNN item-based** [50] - a traditional Collaborative Filtering (CF) approach based on k-nearest-neighbors (KNN) with item-to-item similarity, where reviews are considered as the items. We built a user-review matrix based on the signal that was tested: for the *vote* signal, the entries are represented by the users votes for the reviews (rating score on a scale of 0 to 5). In the *write* signal, the entries are represented in a binary form (1 if the user authored the review, 0 if not). For *both* signal, we converted the votes from numeric to binary: votes in the range of 0-2 were converted to 0, and votes in the range of 3-5 were converted into 1. Then, it combined together with the *write* signal data. To find nearest neighbors we used cosine similarity between reviews vectors and calculated the predicted rating scores for each user and review. Then, we ranked the product reviews for a user based on the

[2]The paper did not provide a publicly available code for the model, thus we did our best effort to implement the model according to the description provided in the paper.

predicted rating scores in descending order. We implemented the model with Surprise [3] and performed hyper-parameter tuning, which included adjusting the minimum common k, the maximum k neighbors, and the minimum support similarity.

- **KNN user-based** - a CF approach similar to KNN item-based, but here to find nearest neighbors the cosine similarity was calculated between users vectors. Then, the predicted rating scores of each user and review was used to rank the product reviews for a user with the scores sorted in descending order.

7.1.4 *Deep learning methods.*

- **DeepFM** [15] - a hybrid model that combines the factorization machines (FM) and deep neural networks (DNN) for recommendation tasks. The FM component captures low-order interactions, while the DNN component models complex, high-order interactions among features. This is a state-of-the-art algorithm for solving binary classification problems like click prediction. We used the implementation of DeepCTR [5] with a regression task and Adam optimizer. We performed hyper-parameter tuning, which included adjusting the embedding dimension, number of hidden units, L2 regularization, and dropout rate.
- **NRMS** [51] - a method that utilizes multi-head self-attention networks together with additive attention to learn news and user representations. The news representations are based on the news title and the user representations on browsing history of the user. For our experiments, we used the news encoder as a review encoder that gets as input the review text with a maximum length of 350 tokens. The user encoder used a "history" of up to 10 reviews, in our case the history refers to reviews the user interacted with based on the user's *signal-based review set*. The model trained using a negative sampling technique. For each review the user interacted with (regarded as a positive sample), we randomly sample K reviews the user did not vote for or did not author. Our implementation is based on NRMS-Pytorch [4]. We used Adam optimizer and performed hyper-parameter tuning for the learning rate, weight decay, batch size, and dropout rate.

7.2 Dataset Preparation

The ground truth for the ranking of product reviews for each user was determined by the actual rating scores (from 0 to 5) they assigned to the reviews they voted for. The reviews were sorted in descending order based on these scores. We handle reviews with no votes as follows - we categorize them to three groups: positive, negative or neutral. If for a given product the user vote only for positive reviews, the other reviews were considered as negative and were placed below the positive reviews. If the user vote only for negative reviews, the other reviews were considered as positive and were placed above the negative reviews. If the user vote for both negative and positive reviews, the remaining reviews were considered as neutral and were placed between the groups. To create the train, validation and test sets, we split the 4-core dataset according to the user voting data (where each user can vote to one or more product reviews). For each user, we created a list of products they interacted with by considering the reviews they voted for. We then split the products into 60%, 20%, and 20% for the train, validation, and test sets, respectively. In case the user voted for exactly one product review, we included the product in the train set.

7.3 Metrics

We use three common metrics to evaluate the models performance:

- **Normalized Discounted Cumulative Gain (NDCG)**: measures ranking quality in recommendations and information retrieval systems. It takes into account the rank position information and the actual value of the rating. NDCG calculated as:

$$NDCG@K = \left(\sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)} \right) / \, IDCG@K \tag{1}$$

  where rel_i is the relevance rating score of the item at position i. IDCG refer to ideal DCG representing the ideal order of ranking.
- **Recall**: measures the proportion of correctly identified relevant items in the top-K recommendations out of the total number of relevant items. Recall is calculated as follows:

$$Recall@K = \frac{\text{Number of relevant items in top-k}}{\text{Total number of relevant items}} \tag{2}$$

- **Hit Ratio**: measures the presence of at least one relevant recommendation, out of the available items. It is calculated as follows:

$$Hit@K = \frac{\text{Number of relevant items in top-k}}{\text{Total number of available items}} \tag{3}$$

We choose Hit Ratio in addition to Recall to assess whether we successfully recommended at least one review the user liked within the top-k. We report the metrics for top-k recommendations and focus on up to 5 reviews, as we believe this is a reasonable number of reviews a user is likely to read on a product page.

8 RESULTS

In this section, we present the results of our proposed approach for the personalized review recommendation task. Table 6 depicts the results of 12 different models across five categories. The table divided into five sections according to models types: basic non-personalized methods, content-based, collaborative filtering, deep learning, and graph-based models which includes our suggested *MAGLLM* method. For each category, the models tested with the three signals—*vote*, *write*, and *both*. We report the results using NDCG@5, Recall@1,5, and Hit@1,5 metrics.

Voting vs. Authorship. The results demonstrate that the *vote* signal consistently outperforms the *write* signal in vast majority of the tested models (except for Games in the Top Terms method, and DVDs in the NRMS method) leading to improved personalization. In some cases, combining both signals further boosts performance, particularly in CF methods and DeepFM. These findings suggests that the voting signal provides an advantage over the authorship signal and more accurately captures user preferences.

Methods Performance. The CF methods achieve higher performance than the content-based methods as is commonly observed in other studies for non-extremely sparse scenarios [8]. The popularity baseline, as observed in previous studies [22], is notably strong and often outperforms content-based, CF, and even more advanced models like DeepFM. However, more complex models such as NRMS and graph-based methods, which capture the intricate relationships between all involved entities, are able to surpass the popularity baseline. This highlights the value of methods with richer structures that capture detailed and diverse information for the personalization task. The graph-based methods, including only graph (MAG)

Table 6: Models performance. The best signal result in each model is boldfaced. The best result in a column is underlined.

Category		DVDs					Beauty					Food & Drink					Internet					Games				
Baseline	Signal	N@5	R@1	R@5	H@1	H@5	N@5	R@1	R@5	H@1	H@5	N@5	R@1	R@5	H@1	H@5	N@5	R@1	R@5	H@1	H@5	N@5	R@1	R@5	H@1	H@5
Non-Personalized																										
Random		0.397	11.25%	53.77%	16.41%	60.97%	0.447	13.77%	65.53%	18.17%	70.29%	0.411	12.49%	57.91%	17.11%	64.03%	0.336	9.36%	42.60%	14.00%	49.32%	0.413	11.90%	56.95%	17.12%	62.59%
Popularity		0.583	25.84%	72.68%	34.16%	79.31%	0.620	29.37%	81.10%	36.22%	85.12%	0.562	24.89%	72.23%	32.20%	78.02%	0.545	24.78%	63.97%	32.90%	72.05%	0.613	30.46%	75.36%	38.42%	80.35%
Content Based																										
Top Terms	like	**0.447**	**14.47%**	**59.31%**	**20.48%**	**66.68%**	**0.501**	17.90%	70.97%	23.03%	**75.70%**	0.440	14.15%	61.51%	19.21%	67.67%	0.386	11.66%	48.37%	17.14%	56.05%	0.453	15.12%	60.84%	20.99%	66.62%
	write	0.442	14.10%	58.84%	19.99%	66.17%	0.492	16.78%	70.61%	21.67%	75.37%	0.434	13.80%	60.91%	18.77%	67.11%	0.384	11.73%	48.25%	16.96%	55.83%	**0.472**	**15.92%**	**63.36%**	**21.73%**	**68.94%**
	both	**0.447**	14.45%	59.27%	20.47%	66.65%	**0.501**	**17.94%**	**70.98%**	**23.06%**	75.66%	**0.441**	**14.17%**	**61.62%**	**19.25%**	**67.79%**	**0.387**	**11.76%**	**48.46%**	**17.28%**	**56.13%**	0.453	15.17%	60.77%	21.06%	66.52%
Word2Vec	like	**0.485**	**17.20%**	**63.37%**	**23.77%**	**70.69%**	**0.532**	**20.67%**	**73.86%**	**26.14%**	**78.54%**	**0.470**	**16.63%**	64.04%	**22.23%**	70.22%	**0.426**	**14.27%**	**52.66%**	**20.54%**	**60.52%**	**0.506**	19.53%	65.68%	26.18%	71.43%
	write	0.467	15.71%	61.53%	21.99%	68.88%	0.513	18.80%	72.38%	24.03%	77.06%	0.457	15.30%	63.01%	20.77%	69.14%	0.416	13.79%	51.67%	19.79%	59.58%	0.491	17.24%	65.55%	23.39%	71.16%
	both	**0.485**	17.09%	63.36%	23.66%	**70.69%**	**0.532**	20.66%	73.75%	26.13%	78.44%	**0.470**	16.60%	**64.08%**	22.21%	**70.28%**	**0.426**	**14.27%**	**52.67%**	**20.54%**	**60.52%**	**0.506**	**19.59%**	**65.70%**	**26.25%**	**71.47%**
Sentence-BERT	like	**0.469**	**16.44%**	**61.20%**	**22.96%**	**68.56%**	**0.571**	**20.16%**	**75.18%**	**29.21%**	**82.66%**	**0.458**	**15.72%**	**62.80%**	**21.14%**	**69.07%**	**0.406**	12.96%	**50.73%**	18.72%	**58.50%**	0.472	**16.71%**	62.25%	**22.95%**	68.09%
	write	0.460	15.62%	60.54%	21.91%	67.86%	0.502	17.89%	71.30%	22.93%	76.06%	0.449	14.72%	62.12%	20.05%	68.34%	0.399	12.87%	49.92%	18.40%	57.54%	**0.477**	16.27%	**64.04%**	22.15%	**69.64%**
	both	0.468	16.42%	61.17%	22.94%	68.53%	0.514	19.29%	72.01%	24.58%	76.75%	**0.458**	15.63%	62.72%	21.03%	69.00%	**0.406**	**13.03%**	50.70%	**18.80%**	58.45%	0.472	16.69%	62.35%	22.92%	68.19%
A2SPR	like						0.519	18.49%	73.84%	23.67%	78.16%											0.504	18.74%	67.55%	24.69%	72.28%
	write						0.499	17.60%	71.14%	22.59%	75.75%											0.483	17.08%	64.59%	22.90%	70.04%
	both						**0.534**	**18.97%**	**75.64%**	**24.22%**	**80.22%**											**0.546**	**21.54%**	**70.90%**	**28.27%**	**76.69%**
Collaborative Filtering																										
KNN item-based	like	**0.510**	**19.74%**	**65.51%**	**26.53%**	**72.59%**	0.556	23.39%	75.37%	29.28%	79.89%	0.471	16.12%	65.05%	21.69%	70.76%	0.466	18.59%	**56.18%**	25.05%	**63.89%**	0.539	22.93%	68.67%	29.96%	74.14%
	write	0.404	11.43%	54.80%	16.84%	61.91%	0.452	14.41%	65.83%	19.08%	70.61%	0.414	12.43%	58.43%	17.10%	64.42%	0.344	9.37%	43.86%	14.09%	50.81%	0.417	12.70%	56.81%	18.17%	62.41%
	both	0.508	**19.74%**	65.42%	26.13%	72.17%	**0.641**	**32.19%**	**82.18%**	**38.93%**	**86.33%**	**0.559**	**24.24%**	**72.73%**	**30.69%**	**78.27%**	**0.469**	**19.21%**	56.11%	**25.79%**	63.30%	**0.578**	**28.24%**	**70.77%**	**35.55%**	**76.03%**
KNN user-based	like	**0.530**	**21.84%**	**66.82%**	**28.70%**	**74.02%**	0.589	26.92%	77.54%	32.84%	82.01%	0.527	20.90%	70.08%	26.79%	76.01%	**0.468**	**18.94%**	56.11%	**25.25%**	**63.90%**	0.512	22.49%	63.86%	29.53%	69.56%
	write	0.403	11.39%	54.84%	16.68%	62.00%	0.452	14.09%	66.12%	18.54%	71.08%	0.411	12.24%	58.19%	16.81%	64.42%	0.347	9.59%	44.18%	14.43%	51.18%	0.411	11.93%	56.57%	17.39%	62.10%
	both	0.482	16.76%	63.85%	22.68%	70.94%	**0.606**	**26.96%**	**80.93%**	**33.00%**	**85.26%**	**0.564**	**23.94%**	**73.83%**	**30.36%**	**79.49%**	0.465	18.07%	**56.41%**	24.37%	63.87%	**0.542**	**24.76%**	**67.68%**	**31.07%**	**73.23%**
Deep Learning																										
DeepFM	like	0.508	18.53%	66.50%	25.24%	73.44%	0.558	22.08%	76.91%	27.90%	81.27%	0.485	17.15%	66.45%	23.04%	72.35%	0.474	17.52%	58.69%	23.95%	66.37%	0.547	22.90%	**70.07%**	**30.09%**	75.37%
	write	0.442	13.90%	58.93%	19.71%	66.14%	0.473	16.26%	67.65%	21.26%	72.46%	0.431	13.80%	60.02%	18.91%	66.07%	0.341	9.24%	43.21%	14.18%	50.20%	0.419	13.17%	56.73%	18.81%	62.29%
	both	**0.525**	**19.78%**	**68.11%**	**26.64%**	**75.16%**	**0.596**	**26.26%**	**79.56%**	**32.65%**	**83.88%**	**0.519**	**20.03%**	**69.37%**	**26.28%**	**75.22%**	**0.496**	**21.03%**	**58.71%**	**28.10%**	**66.62%**	**0.554**	**24.50%**	69.80%	20.06%	**75.38%**
NRMS	like	0.681	36.35%	80.59%	46.06%	86.66%	**0.731**	**42.54%**	**88.71%**	**50.68%**	**92.21%**	**0.696**	**37.56%**	**84.56%**	**46.58%**	**89.38%**	0.604	29.85%	**69.35%**	39.15%	**77.59%**	**0.647**	**33.23%**	**77.94%**	**41.70%**	**83.48%**
	write	0.692	37.67%	**81.34%**	47.64%	**87.27%**	0.724	41.31%	88.50%	49.51%	91.90%	0.690	36.59%	84.28%	45.82%	88.99%	0.596	29.36%	68.46%	38.73%	76.82%	0.598	27.74%	74.38%	35.49%	79.91%
	both	**0.693**	**37.83%**	81.23%	**47.74%**	87.20%	0.725	41.54%	88.43%	49.76%	91.90%	0.694	37.35%	84.44%	46.52%	89.17%	**0.606**	**30.47%**	68.89%	**39.83%**	77.33%	0.615	30.49%	75.16%	38.51%	80.64%
Graphs																										
MAG	like	**0.876**	**70.42%**	**88.80%**	**84.89%**	**90.59%**	**0.898**	**74.74%**	**92.58%**	**86.22%**	**93.54%**	0.837	67.93%	85.98%	80.78%	87.08%	**0.881**	**69.74%**	**87.14%**	**85.89%**	**90.04%**	0.882	71.65%	**89.94%**	83.54%	**92.24%**
	write	0.704	45.45%	80.38%	54.56%	84.89%	0.689	43.21%	84.07%	49.93%	87.11%	0.664	41.11%	80.01%	48.32%	83.67%	0.604	34.72%	68.97%	42.55%	75.20%	0.723	52.13%	78.65%	61.34%	82.05%
	both	0.856	67.67%	87.22%	81.89%	89.18%	0.865	71.47%	89.84%	82.76%	90.77%	**0.843**	**68.34%**	**86.70%**	**81.25%**	**87.86%**	0.862	67.81%	85.67%	83.66%	88.34%	**0.883**	**72.00%**	89.69%	**84.65%**	91.81%
MAGLLM	like	**0.883**	**71.29%**	**89.29%**	**85.39%**	**91.15%**	**0.911**	**76.96%**	**93.29%**	**88.38%**	**94.23%**	0.858	69.41%	88.46%	82.13%	89.75%	**0.894**	**70.65%**	**89.46%**	**86.64%**	**92.30%**	**0.912**	**75.48%**	**92.07%**	**87.91%**	**94.29%**
	write	0.669	42.60%	77.12%	51.73%	81.07%	0.680	41.04%	84.17%	47.40%	87.30%	0.639	40.01%	76.91%	47.44%	79.95%	0.616	39.59%	67.72%	50.00%	71.70%	0.745	54.17%	79.84%	64.53%	83.35%
	both	0.850	68.44%	86.20%	82.26%	87.94%	0.875	72.08%	90.93%	83.20%	91.93%	**0.863**	**70.40%**	**88.69%**	**83.17%**	**89.91%**	0.857	65.75%	86.70%	81.23%	89.88%	0.899	74.87%	90.26%	87.20%	92.39%

and our proposed approach *MAGLLM* which combined graph with LLM, achieve the highest performance across all categories when using the *vote* or *both* signals. Specifically, our method *MAGLLM* outperforms all other methods across all categories. In addition, the graph methods present the largest performance gap between the *write* and *vote* signals. This is likely due to the richer information represented in the graph and the ability of meta-paths to capture complex relationships between nodes. For example, looking at relationship between two users, the meta-path *Voter-Review-Voter* (VRV) can be formed because two users can vote for the same review, whereas *Author-Review-Author* (ARA) cannot be formed, as two users cannot be considered as authors of the same review. In most cases, our method *MAGLLM* successfully ranked at least one review the user liked within the top-5 reviews, as the Hit@5 metric reaches at least 90% across all categories.

9 DISCUSSION

We presented, for the first time in review personalization, the use of voting signals. We compare the use of review authorship and review voting signals for personalization across popular recommendation methods and five different e-commerce domains. Our results indicate that using the voting signal consistently yields substantially higher personalization performance over the authorship signal. In some cases, combining the authorship and voting signals yields additional improvement over using the voting signal alone. We also suggested *MAGLLM*, a personalized review recommendation technique that models relationships among products, reviews, and users using heterogeneous graph network with meta-paths and enhances user representation using LLM. *MAGLLM* reaches high results with Hit@5 varying from 89.9% (Food & Drink) to 94.2% (Beauty and Games) that allow high quality personalization.

The performance improvement from using the voting signal indicates its significant potential for enhancing personalized recommendations. Given these findings, we suggest several important practical implications. First, make the voting feature visible to encourage more voting. It is crucial to consider the role of the voting option for reviews on e-commerce platforms. Some platforms may not invest enough in this feature or may not provide it at all, but it can be a powerful tool in improving recommendation systems in various contexts. Second, consider offering more elaborate voting options, such as rating on a star scale or multi-dimensional feedback like on Yelp (e.g., helpful, funny, or interesting). These forms of detailed feedback would make it easier to capture nuanced user preferences, ultimately leading to more accurate and effective personalization. Third, give prominence also to the visibility of aggregate votes per review. Highlighting how many users have found a review helpful or engaging can motivate others to contribute their feedback, creating increased user interaction and more reliable personalization data. Forth, encouraging user feedback is essential. While writing a review is often time-consuming and requires more effort, providing direct feedback through voting mechanisms, such as likes, dislikes, or numeric ratings, is much simpler and more intuitive for many users. This ease of engagement allows for a broader spectrum of user participation. Finally, as generative AI becomes more prevalent, automated review generation is likely to become more popular. While this may reduce the need for users to write full reviews, voting will remain a valuable form of explicit feedback. Therefore, investing in this feature could be highly beneficial.

For future work, we plan to experiment with other graph-based approaches to further explore the potential of heterogeneous graphs in review recommendation. Additionally, conduct in-vivo experiments within live environments could offer valuable insights into the real-world applicability of our approach. This would allow us to measure the impact of the voting signal in dynamic, online systems.

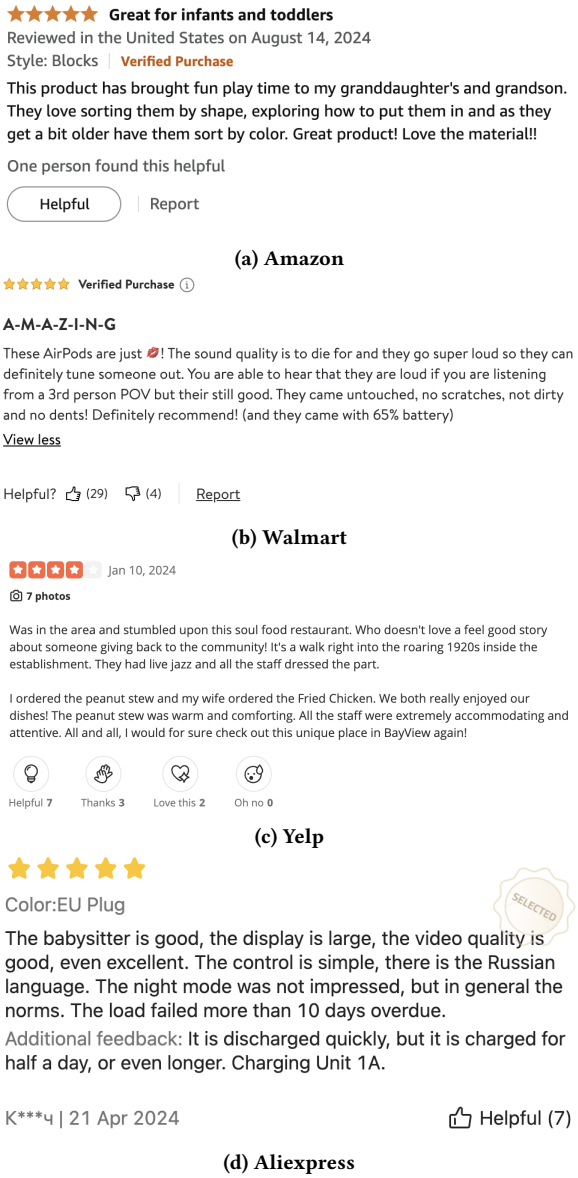
REFERENCES

[1] 2024. MAGNN. https://github.com/cynricfu/MAGNN/tree/master
[2] 2024. Python bindings. https://github.com/abetlen/llama-cpp-python
[3] 2024. A Python scikit for building and analyzing recommender systems. https://github.com/NicolasHug/Surprise
[4] 2024. Pytorch Implementation of EMNLP 2019 NRMS. https://github.com/aqweteddy/NRMS-Pytorch
[5] 2024. PyTorch package of deep-learning based CTR models. https://github.com/shenweichen/DeepCTR-Torch
[6] 2024. Sentires: A Toolkit for Phrase-level Sentiment Analysis. https://github.com/evison/Sentires
[7] Deepak Agarwal, Bee-Chung Chen, and Bo Pang. 2011. Personalized recommendation of user comments via factor models. In *Proceedings of the 2011 conference on empirical methods in natural language processing*. 571–582.
[8] Parul Aggarwal, Vishal Tomar, and Aditya Kathuria. [n. d.]. Comparing content based and collaborative filtering in recommender systems. *International Journal of New Technology and Research* 3, 4 ([n. d.]), 263309.
[9] Wenshuo Chao, Zhi Zheng, Hengshu Zhu, and Hao Liu. 2024. Make Large Language Model a Better Ranker. *arXiv:2403.19181* (2024).
[10] Fermín L Cruz, José A Troyano, Fernando Enríquez, F Javier Ortega, and Carlos G Vallejo. 2013. 'Long autonomy or long delay?'The importance of domain in opinion mining. *Expert Systems with Applications* 40, 8 (2013), 3174–3184.
[11] Anupam Dash, Dongsong Zhang, and Lina Zhou. 2021. Personalized ranking of online reviews based on consumer preferences in product features. *International Journal of Electronic Commerce* 25, 1 (2021), 29–50.
[12] Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805* (2018).
[13] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.
[14] Kavita Ganesan and ChengXiang Zhai. 2012. Opinion-based entity ranking. *Information retrieval* 15, 2 (2012), 116–150.
[15] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv:1703.04247* (2017).
[16] Lei Guo, Enhua Tan, Songqing Chen, Xiaodong Zhang, and Yihong (Eric) Zhao. 2009. Analyzing patterns of user content generation in online social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. 369–378.
[17] Ido Guy. 2022. Social Recommender Systems. *Recommender Systems Handbook* (2022), 835–870.
[18] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*. Springer, 364–381.
[19] Chunli Huang, Wenjun Jiang, Jie Wu, and Guojun Wang. 2020. Personalized review recommendation based on users' aspect sentiment. *ACM Transactions on Internet Technology (TOIT)* 20, 4 (2020), 1–26.
[20] Reda Igebaria, Eran Fainman, Sarai Mizrachi, Moran Beladev, and Fengjun Wang. 2024. Enhancing Travel Decision-Making: A Contrastive Learning Approach for Personalized Review Rankings in Accommodations. *arXiv:2407.00787* (2024).
[21] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. 1749–1752.
[22] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A Re-visit of the Popularity Baseline in Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1749–1752. https://doi.org/10.1145/3397271.3401233
[23] Farhad Khalilzadeh and Ilyas Cicekli. 2024. REHREC: Review Effected Heterogeneous Information Network Recommendation System. *IEEE Access* 12 (2024), 42751–42760.
[24] Soo-Min Kim, Patrick Pantel, Timothy Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on empirical methods in natural language processing*. 423–430.
[25] Theodoros Lappas, Mark Crovella, and Evimaria Terzi. 2012. Selecting a characteristic set of reviews. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 832–840.
[26] Huiting Liu, Yi Chen, Peipei Li, Peng Zhao, and Xindong Wu. 2023. Enhancing review-based user representation on learned social graph for recommendation. *Knowledge-Based Systems* 266 (2023), 110438.
[27] Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint* cs/0205028 (2002).
[28] Tianhui Ma, Yuan Cheng, Hengshu Zhu, and Hui Xiong. 2023. Large Language Models are Not Stable Recommender Systems. *arXiv preprint arXiv:2312.15746* (2023).
[29] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 142–150.
[30] Muhammad Faraz Manzoor, Adnan Abid, Naeem A Nawaz, and Atif Alvi. 2022. Aspect based sentence segregated dataset of hybrid car's consumers online reviews. *Data in Brief* 42 (2022), 108293.
[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781* (2013).
[32] Samaneh Moghaddam, Mohsen Jamali, and Martin Ester. 2011. Review recommendation: personalized prediction of the quality of online reviews. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 2249–2252.
[33] Samaneh Moghaddam, Mohsen Jamali, and Martin Ester. 2012. Etf: extended tensor factorization model for personalizing prediction of review helpfulness. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 163–172.
[34] Andreas Munzel and Werner H. Kunz. 2014. Creators, multipliers, and lurkers: who contributes and who benefits at online review sites. *Journal of Service Management* 25, 1 (2014), 49–74.
[35] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*. 188–197.
[36] Akhil Sai Peddireddy. 2020. Personalized Review Ranking for Improving Shopper's Decision Making: A Term Frequency based Approach. *arXiv:2009.03258* (2020).
[37] Chee Wei Phang, Atreyi Kankanhalli, and Bernard CY Tan. 2015. What motivates contributors vs. lurkers? An investigation of online feedback forums. *Information Systems Research* 26, 4 (2015), 773–792.
[38] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv:1908.10084* (2019).
[39] Gang Ren, Lei Diao, Fanjia Guo, and Taeho Hong. 2024. A co-attention based multi-modal fusion network for review helpfulness prediction. *Information Processing & Management* 61, 1 (2024), 103573.
[40] Sunil Saumya, Jyoti Prakash Singh, and Yogesh K Dwivedi. 2020. Predicting the helpfulness score of online reviews using convolutional neural network. *Soft Computing* 24, 15 (2020), 10989–11005.
[41] Jie Shuai, Kun Zhang, Le Wu, Peijie Sun, Richang Hong, Meng Wang, and Yong Li. 2022. 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*. 1283–1293.
[42] Na Sun, Patrick Pei-Luen Rau, and Liang Ma. 2014. Understanding lurkers in online communities: A literature review. *Computers in Human Behavior* 38 (2014), 110–117.
[43] Vaishak Suresh, Syeda Roohi, Magdalini Eirinaki, and Iraklis Varlamis. 2014. Using Social Data for Personalizing Review Rankings.. In *RSWeb@ RecSys*.
[44] Lei Tan, Daofu Gong, Jinmao Xu, Zhenyu Li, and Fenlin Liu. 2023. Meta-path fusion based neural recommendation in heterogeneous information networks. *Neurocomputing* 529 (2023), 236–248.
[45] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 93–102.
[46] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv:2302.13971* (2023).
[47] Panayiotis Tsaparas, Alexandros Ntoulas, and Evimaria Terzi. 2011. Selecting a comprehensive set of reviews. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 168–176.
[48] Bingkun Wang, Yulin Min, Yongfeng Huang, Xing Li, and Fangzhao Wu. 2013. Review rating prediction based on the content and weighting strong social relation of reviewers. In *Proceedings of the 2013 international workshop on Mining unstructured big data using natural language processing*. 23–30.
[49] Haiming Wang, Wei Liu, and Jian Yin. 2021. Multi-Task Learning with Personalized Transformer for Review Recommendation. In *WISE*. Springer, 162–176.
[50] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*. 501–508.
[51] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019. Neural news recommendation with multi-head self-attention. In *EMNLP-IJCNLP*. 6389–6394.
[52] Yelp. 2024. Yelp Dataset. https://www.yelp.com/dataset
[53] Yijiong Yu, Huiqiang Jiang, Xufang Luo, Qianhui Wu, Chin-Yew Lin, Dongsheng Li, Yuqing Yang, Yongfeng Huang, and Lili Qiu. 2024. Mitigate Position Bias in Large Language Models via Scaling a Single Dimension. *arXiv:2406.02536* (2024).
[54] Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. LlamaRec: Two-stage recommendation using large language models for ranking. *arXiv:2311.02089* (2023).

10 APPENDIX

10.1 Screenshots of review feedback functionality on various e-commerce platforms.

Figure 3 demonstrates the review feedback functionality across four popular e-commerce platforms. It can be seen that there are subtle differences between the platforms, where the review feedback can be a simple indication as "helpful" (Amazon, Aliexpress), a thumbs up or thumbs down for "liking" or "disliking" (Walmart), or a multi-dimensional with "helpful", "thanks", "love this", and "oh no" (Yelp).



(a) Amazon

(b) Walmart

(c) Yelp

(d) Aliexpress

Figure 3: Helpful feature in different e-commerce platforms

10.2 LLM Prompts of User Profile

Figure 4 illustrates some of the prompts we used in the experimentation of generating user profile using LLM based on the user history, which is the *signal-based review set* associated with the user.

```
system: You are required to generate user profile based on the history of a
user. The profile should contain only user interests that can be learned
from the given history. Do not infer the user name, age or gender.
The profile will later be used to calculate personalized recommendation of
new reviews. Use up to 300 tokens.
prompt: The user previously {signal} the following reviews: <history>.
His profile:
```

```
system: Your objective is to create user profile using their review
history. The profile should be general, without any personal details, but
with enough details to allow personalized recommendations of new reviews.
The profile should contain up to 300 tokens.
prompt: The user previously {signal} the following reviews: <history>.
His profile:
```

Figure 4: LLM prompts examples for generating user profile using the user history