Transparent and Scrutable Recommendations Using Natural Language User Profiles

Jerome Ramos[†] **Hossein A. Rahmani**[†] **Xi Wang**[†‡] **Xiao Fu**[†] **Aldo Lipani**[†]
[†]University College London
[‡]The University of Sheffield
{jerome.ramos.20, hossein.rahmani.22, xiao.fu.20, aldo.lipani}@ucl.ac.uk
xi.wang@sheffield.ac.uk

Abstract

Recent state-of-the-art recommender systems predominantly rely on either implicit or explicit feedback from users to suggest new items. While effective in recommending novel options, many recommender systems often use uninterpretable embeddings to represent user preferences. This lack of transparency not only limits user understanding of why certain items are suggested but also reduces the user's ability to scrutinize and modify their preferences, thereby affecting their ability to receive a list of preferred recommendations. Given the recent advances in Large Language Models (LLMs), we investigate how a properly crafted prompt can be used to summarize a user's preferences from past reviews and recommend items based only on language-based preferences. In particular, we study how LLMs can be prompted to generate a natural language (NL) user profile that holistically describe a user's preferences. These NL profiles can then be leveraged to fine-tune a LLM using only NL profiles to make transparent and scrutable recommendations. Furthermore, we validate the scrutability of our user profile-based recommender by investigating the impact on recommendation changes after editing NL user profiles. According to our evaluations of the model's rating prediction performance on two benchmarking rating prediction datasets, we observe that this novel approach maintains a performance level on par with established recommender systems in a warm-start setting. With a systematic analysis into the effect of updating user profiles and system prompts, we show the advantage of our approach in easier adjustment of user preferences and a greater autonomy over users' received recommendations.

1 Introduction

Personalized recommender systems often rely on building latent representations from past user interactions to provide recommendations. These models, while effective, suffer from a lack of interpretability on the learned users' preferences, making it challenging to provide intuitive explanations for recommendations. For example, many Collaborative Filtering (CF) techniques learn latent embeddings (He et al., 2017; Salakhutdinov and Mnih, 2007) to represent information about every user and item. These embeddings are complex, continuous vectors that cannot be easily interpreted or modified (Radlinski et al., 2022). In fact, since collaborative filtering builds embeddings via past interactions, users would need to significantly change their interaction history to receive new recommendations that align with their current preferences.

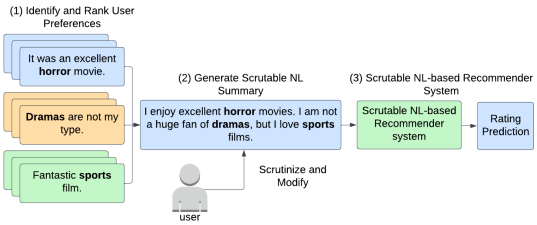

Figure 1: Overall architecture of UPR. In Step 1, we identify and rank user preferences from user reviews. Features are highlighted in **bold**. In Step 2, we use an LLM to generate a personalized, scrutable natural language (NL) profile based on the user's top features. In Step 3, we train a scrutable, NL-based recommender system using the generated NL profile to predict the user's rating for a target item.

Given the strong performance of Large Language Models (LLMs) in a wide variety of NLP tasks (Touvron et al., 2023; Jiang et al., 2023) as well as the inherent scrutable nature of text-based inputs, we explore how we can prompt LLMs with a natural language (NL) description of a user's preferences (i.e., user profiles) to provide greater transparency on how a model provides new recommendations. Furthermore, by using NL preferences rather than embeddings, a user can easily scrutinize and update their preferences when needed.

For recommender systems, we define *trans-*

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13971–13984

parency as a faithful representation behind the recommendation mechanism and *scrutability* as a direct and meaningful way to inspect and modify a user representation (Zhang and Chen, 2020). Addressing these issues is vital: if users receive poor recommendations, they should understand why and have the means to modify their representation for better results. Research has shown that explanations that justify a recommendation help users make better and faster decisions and promote trust in the system (Zhang and Chen, 2020; Tintarev and Masthoff, 2015).

Recent works in transparent recommendation systems have explored using pre-defined templates (Li et al., 2020) and automatically generated text (Li et al., 2021, 2023a; Xie et al., 2022) to explain recommendations to users. However, these works focused on explaining individual item recommendations rather than holistic, personalized preferences. Another approach to transparent and scrutable recommender systems is set-based preferences, which aim to explain a user's *general preferences* for a particular domain (Radlinski et al., 2022; Balog et al., 2019). Specifically, rather than giving a rating for each individual item, a user can categorize groups of items into *sets*, which are then used to recommend new items. For example, a user can state that they generally like 'action' movies and generally dislike 'romance' movies. This transparency lets users understand the model's interpretation and adjust their profile accordingly, leading to more aligned recommendations. Chang et al. (Chang et al., 2015) have shown that set-based preferences allow users to elicit preferences faster, leading to higher satisfaction with the resulting recommended items. Sanner et al. (2023) found that LLMs are competitive recommender systems in a near cold-start setting. In particular, they collected language-based user profiles from crowdsourced workers and found that by using few-shot prompting with the PaLM model (Chowdhery et al., 2022), they were able to achieve similar performance to item-based collaborative filtering methods.

In this paper, we propose **U**ser **P**rofile **R**ecommendation (UPR) depicted in Fig. 1, a language-based approach to set-based, transparent recommendations in a warm-start setting. Unlike a cold-start setting, a warm-start setting contains significantly more user-item interactions and could further improve the performance of collaborative filtering methods (He et al., 2017). In lieu of a real-world dataset of NL profiles, which is both expensive and challenging to collect, we instead simulate users writing and modifying NL profiles by prompting instruction-tuned LLMs to generate profiles based on past user reviews. We first exemplify that these machine-generated NL profiles qualitatively summarize a user's preferences concisely while maintaining a scrutable format. Furthermore, according to the conducted experiments on two benchmark datasets, Amazon Review (Movies and TV) and TripAdvisor , we show how NL user profiles can be used to train a transparent, NL-based recommendation system based on scrutable NL profiles rather than uninterpretable user embeddings. We observe that our NL-based approach has competitive recommendation performance to popular baseline recommender systems while also being transparent and scrutable.

Our model enables a transparent and scrutinized recommendation process with competitive recommendation accuracy. In addition, past work suggests that a transparent and scrutable model is preferred because it allows users to understand the reasoning behind the model's recommendations and easily modify their preferences to receive new recommendations (Radlinski et al., 2022; Balog et al., 2019). To validate the advanced transparency and scrutability of our UPR model, we investigate how editing user profiles affects recommendation performance. For reproducibility, we publicly release all data and code[1].

Our key contributions include:

- A novel method to simulate NL user preference profiles from user feedback by prompting instruction-tuned LLMs, which are shown to be fluent, informative, concise, and relevant through a user study.
- A unique recommendation technique that prioritizes language-based input over learned embeddings, maintaining performance levels comparable to non-transparent baseline recommenders.
- An analysis of how editing a user profile impacts the downstream recommendation task.

[1]https://github.com/jeromeramos70/user-profile-recommendation

2 Related Works

2.1 Explainable Recommendation

Explainable recommendation aims to both generate relevant recommendations and provide a justification for why those items are being recommended. One of the two main perspectives of explainable recommendation research is explainable machine learning (Li et al., 2023a). Explainable machine learning uses various computational techniques to understand and convey why a model returns a particular explanation. Examples of machine-learning-based explainable recommender systems include using pre-defined templates (Li et al., 2020; Balog et al., 2019), counterfactual explanations (Tran et al., 2021), visualization (Geng et al., 2022), and natural language generation for an item-level recommendation, meaning that the explanation is for a target user-item pair (Li et al., 2023a, 2021). Notably, these works (Li et al., 2023a, 2021, 2020) often make use of user and item identifiers to create uninterpretable embeddings, meaning that a user would need to dramatically change their interaction history in order to update their recommendations. Rather than focusing on explaining a single recommendation (i.e., output) for a given user, we focus on making the *input* of the model fully transparent and scrutable. By focusing on natural language as the input of the model, the user has full transparency on what data is used to generate recommendations and can seamlessly update the input as necessary. This approach can be seen as more similar to content filtering methods (Aggarwal, 2016; Deshpande and Karypis, 2004), which matches the attributes of items with the interests or preferences indicated in a user's profile. However, by leveraging LLMs, we are able to use the pretrained knowledge that an LLM has on the item set and fine-tune a user's NL profile to increase performance on the recommendation task.

2.2 Set-Based Preferences

Many papers investigate the scenario where preferences are expressed over sets, which are then used to recommend individual items (Sharma et al., 2019; Balog et al., 2019; Chang et al., 2015). These set-based preferences are then used to generate recommendations for individual items. One example of set-based preferences is tagging, where a user selects keyword(s) for each item. Examples of tags include "superhero", "action", etc. These tags help users indicate their preferences in their own personalized language. Similarly, Mysore et al. (2023a) generates a transparent user profile by selecting a small set of human-readable concepts from a global inventory of keywords to enable controllable recommendations. Past research has shown that users are able to elicit preferences over sets faster than rating each item individually (Chang et al., 2015).

Most similar to our work is the transparent model presented by Sanner et al. (Sanner et al., 2023), which studies how NL profiles can be used as input for LLMs in near-cold start settings to obtain competitive performance with baseline recommender models. The main novelty of our work is that we compare the performance of language-based preferences in a *warm start* scenario, where recommender systems have access to significantly more user-item interactions. Furthermore, rather than collecting human written NL profiles like Sanner et al. (2023), we instead use LLMs to automatically generate an NL profile based on past reviews. Mysore et al. (2023b) has previously shown that historical user reviews can be used to generate synthetic narrative queries with LLMs for narrative-driven recommendation. Not only does this approach allow us to explore how LLMs can be used to summarize user preferences in a scrutable and editable format, but it also mitigates the time-intensive, costly task of crowdsourcing high-quality NL profiles written by users. These NL profiles serve as an automated way to generate set-based preferences from past reviews, which users can modify and edit as needed. Lastly, we are the first work that we are aware of that evaluates how NL profiles can be updated to receive new recommendations from an LLM-based recommender system in a warm-start setting.

3 Methodology

3.1 Goal of User Profiles

Currently, there is no open-source, large-scale dataset of language-based preferences for recommendation. Furthermore, collecting such a dataset is both challenging and expensive because of the amount of data and quality needed to effectively fine-tune and benchmark a model. Instead, we leverage user text reviews and explicit ratings to infer a user's preferences and generate a corresponding NL profile. Not only do these reviews give us insight into which features the user likes and dislikes, but they also allow us to generate personalized profiles written with the user's own terminol-

ogy. In addition, by generating NL profiles, users can automatically receive both their NL profiles and corresponding recommendations. The user can then adjust and modify the profile accordingly, which can save time versus manually crafting a prompt that provides accurate recommendations.

Balog et al. (Balog et al., 2019) explored using tags, user-written keywords to describe an item, to generate template-based profiles. Although tags have been commonly used for recommendation tasks (Bogers, 2018; Chen et al., 2020), past research has pointed out issues with tag quality (Sen et al., 2007). In addition, tags may not provide sufficient granularity of preferences (Radlinski et al., 2022). We argue that text-based profiles are closer to how a user naturally describes their personalized preferences versus a set of tags.

3.2 Identifying and Ranking User Preferences

Let $\mathcal{I}$ be the set of all items with respect to target user u. Each item $i \in \mathcal{I}$ has a (normalized) rating $r \in [-1, 1]$, where ratings lower than 0 mean the user disliked an item and ratings higher than 0 mean the user liked the item. Additionally, each item contains an explanation e, which is a sentence written by the user in their review that best describes their feedback for that item. Using a phrase-level sentiment analysis toolkit (Zhang et al., 2014), we can also extract a feature word f from e, where f is the important keyword of the explanation. For example, in the explanation, "the swimming **pool** is fantastic", pool is the extracted feature. We can then infer a user's set-based preference for a particular feature f by calculating the average rating of $\mathcal{I}_f$, the set of all items whose feature is f:

$$\overline{r_f} = \sum_{i \in \mathcal{I}_f} r_i, \tag{1}$$

where $\overline{r_f} < 0$ means that the user dislikes feature f in general and $\overline{r_f} > 0$ means the user generally likes f. We then calculate a utility score $U(f)$ for a given feature f to take into account how often a user uses a particular feature in reviews and subtract a discount statement based on its statistical strength. We define $U(f)$ as:

$$U(f) = |\overline{r}(f)| \times cov(f) \times sig(f), \tag{2}$$

where $cov(f)$ and $sig(f)$ are the coverage and significance scores for a particular feature f. We define the coverage score as the ratio of items whose feature is f over the entire item set.

$$cov(f) = \frac{|\mathcal{I}_f|}{|\mathcal{I}|}. \tag{3}$$

Additionally, we define the significance score as a discount statement where the statistical strength is less than two standard errors, that is:

$$sig(f) = min\left(2, |\overline{r_f}| / \frac{\sigma_t}{\sqrt{|\mathcal{I}_f|}}\right), \tag{4}$$

where σ_f is the variance of R_f, which is defined as the set of all ratings $r \in R$ with feature f. To reduce the noise in the ranking process, we grouped features by their stems, which were extracted using the NLTK Porter stemmer (Bird et al., 2009). By doing so, we can treat features such as "pool" and "pools" as a single entity rather than two separate features. We manually inspect the preprocessed data to ensure that the grouped entities are similar and remove features that are overly generic (e.g., the feature 'film' in the movie domain). Once the utility scores for the entire dataset have been calculated, we use the features with the highest scores to describe a user's preferences. Since we calculate significance scores using absolute values, features that are *disliked* by the user may also be included in the profile.

Note that our user preference identification and ranking algorithm is similar to what was developed by Balog et al. (Balog et al., 2019) for the MovieLens-20M (ML-20M) dataset (Harper and Konstan, 2015). However, there are a few key differences in our method, which mainly stem from how the user feedback data is structured. For example, our user review data contains only one feature per item rather than multiple tags per item. Thus, our coverage score rewards features that appear more in the dataset rather than adding a penalty if they are overused. In our scenario, we argue that this is beneficial because if a user writes a review with a certain feature often, it means that this feature is important to them.

3.3 Transforming Preferences to NL Profiles

After identifying a user's top features, we can transform the list of relevant reviews into an NL profile that captures a user's general preferences for the target domain. Using an instruction-tuned LLM, we can prompt the model to summarise the list of reviews to create a holistic profile that explains the user's general preferences for a particular domain

(shown in Appendix A.1). After several iterations of prompt engineering, we provide 5 reviews per feature in the prompt, which fits within the max token constraint during experimentation. Since user-written reviews are used in the prompt, the resulting NL profile is generated with each user's personalized semantics. Since there is no ground truth for NL profiles, we instead analyze the quality of the profiles in Section 5.3 and the utility of the generated profiles by evaluating the performance of a transparent recommender system trained on the NL profiles.

3.4 Recommendation Task

Similar to the P5 prompts for LLM-based recommendation (Geng et al., 2022), we complete the rating prediction task by adding the NL profile to a prompt (shown in Appendix A.2) and fine-tuning an LLM to complete a regression task. One key difference from traditional approaches is that no user identifiers or item identifiers are used during the NL profile generation or recommendation tasks. Thus, users can easily scrutinize the recommendation pipeline at any stage because all inputs are natural language-based rather than identifier-based. We further explore scrutability in Section 5.4.

4 Experimental Setup

4.1 Datasets

We conduct our experiments on two publicly available explainable recommender datasets, Amazon Movies and TV (Amazon-MT) (He and McAuley, 2016) and TripAdvisor[2] (hotels), which focus on generating explanations for a user-item pair (Li et al., 2020). Each dataset consists of a user identifier, an item identifier, a rating from 1 to 5, and an explanation extracted from user reviews. We use an 8:1:1 train/validation/test split. The datasets were preprocessed to exclude any records where the title of the item is missing. To ensure a warm-start setting, every user and item appear at least once in each split, and there are at least 5 reviews per user in the training set. For reproducibility, we save all profiles generated for recommendation.

4.2 Evaluation Metrics

To measure recommendation accuracy, we use Root Mean Squared Error (**RMSE**) and Mean Absolute Error (**MAE**), two common metrics in the rating prediction task for these datasets (Li et al., 2020, 2023a). MAP with a relevance threshold of 4.0 and nDCG are calculated using condensed lists, following the evaluation strategy outlined by Sakai (Sakai, 2007); that is, we only consider items that are explicitly rated in the test set.

[2] https://www.tripadvisor.com

Table 1: Statistics of the Amazon-MT and TripAdvisor datasets.

	Amazon-MT	**TripAdvisor**
#users	4,811	9,765
#items	5,459	6,280
#records	288,693	320,023
#features	5,114	5,069
#records per user	60.01	32.77
#records per item	52.88	50.96
#words per explanation	14.14	13.01

4.3 Baselines

We compare our model with the following baseline recommender systems. All baseline models are implemented using the Cornac library with default parameters (Truong et al., 2021).

- **Most Popular**: An algorithm that recommends items with the most ratings.
- **UserKNN-BM25** (Aggarwal, 2016): A collaborative filtering method that uses K-Nearest Neighbors to calculate distance between users with BM25 re-weighting.
- **Item-KNN-cosine** (Deshpande and Karypis, 2004): A collaborative filtering method that uses K-Nearest Neighbors to calculate cosine distance between items.
- **MF** (Koren et al., 2009): Matrix Factorization learns latent user and item embeddings to provide recommendations.
- **NeuMF** (He et al., 2017): a hybrid recommendation model that combines matrix factorization with multi-layer perceptrons.

In addition, we compare our model with two item-level explainable recommender models that describe why a user might like a target item. For example, given "user A" and "item 123" as input, the model might return a rating of 4.0 with the following explanation: "the swimming pool is fantastic". Note that both of these models are considered collaborative filtering, meaning that they mainly rely

on user and item identifiers to generate recommendations. While not directly analogous due to our task of generating explainable NL profiles as input to a recommender model, this comparison yields insights into the performance of explainable models trained for rating prediction.

- **PETER+** (Li et al., 2021): a small transformer designed for rating prediction and explanation generation using context prediction.
- **PEPLER-MLP** (Li et al., 2023a) An item-level explainable model that fine-tunes GPT-2 for rating prediction and explanation generation for a user-item pair.

4.4 Implementation Details

For NL profile generation, we use the features defined in the datasets to identify and rank a user's set-based preferences. We then use a custom prompt for each user to generate an NL profile. We experiment with generating NL profiles using the 7B instruction-tuned versions of Llama2 (Touvron et al., 2023) and Mistral (Jiang et al., 2023). We generate each NL profile with a maximum of 200 tokens, a number of features set to 5, and a temperature of 0.7. For reproducibility, we save all generated user profiles used to fine-tune our model[3].

For the recommendation task, we train each model on the training set, tune hyperparameters on the validation set, and report the performance on the test set. The models are trained using either the Llama2 or Mistral-based profiles. We use GPT-2 (Radford et al., 2019) as our pre-trained language model and train with a batch size of 8 and the AdamW optimizer (Kingma and Ba, 2015) with a learning rate of 3e-4. For full implementation details, including prompts used and hyperparameter tuning, refer to Appendix A.

5 Results and Analysis

5.1 Recommendation Performance

We report the performance of our model compared to baseline models in Table 2. We observe that UPR performs comparatively well with our baseline models and is consistently competitive across all metrics. Importantly, all compared models learn user preferences from *all* interactions by taking advantage of the efficiency of embeddings. In contrast, UPR is limited to the information provided in the profile, which has a maximum length of 200 tokens. Consequently, UPR can only learn from a fraction of the interactions in the dataset. Nevertheless, UPR performs fairly well under the aforementioned constraints, meaning that LLMs can identify relevant items using only interaction data distilled into a short amount of text.

Even though the NL profile comparatively contains fewer features about the user, we observe that scrutable NL profiles can be used as a substitute for uninterpretable user embeddings without sacrificing much in performance. Given that newer models such as GPT-3 (Brown et al., 2020) have shown better performance than GPT-2 (Radford et al., 2019) and are trained on more data and natural language tasks, we anticipate that our model's performance will improve with larger-scale models. However, SOTA models contain significantly more parameters, meaning that more powerful hardware is needed to train them.

Given the competitive performance of UPR, we argue that an explainable model may be useful in real-life scenarios where human users value the transparency and scrutability of a set-based preference model. For example, if a user wanted to change their profile preferences in the collaborative filtering model, they would need to make many changes to their interaction history to see a substantial difference in their recommended items. For future work, we plan to study how human users interact with UPR and investigate the tradeoffs between traditional recommender methods and a transparent, scrutable recommender model.

5.2 Effects of Number of Features on Recommendation Performance

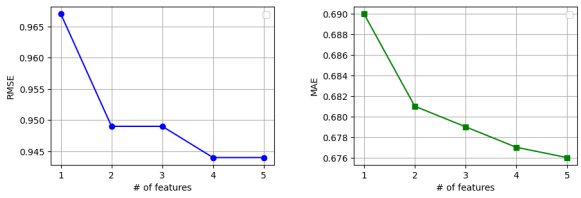

Figure 2: RMSE and MAE over a varying number of features in NL profile for the **Amazon-MT** dataset.

To examine the impact that the number of features contained in an NL profile has on recommendation performance, we run an ablation study that studies how changing k, the number of features used to generate the NL profile affects the down-

[3]https://github.com/jeromeramos70/user-profile-recommendation

Table 2: Recommendation Performance on the Amazon-MT and TripAdvisor datasets. The best-performing values are highlighted in **bold**.

	Amazon-MT				TripAdvisor			
Model	RMSE	MAE	nDCG@10	MAP	RMSE	MAE	nDCG@10	MAP
MostPop	1.505	0.962	0.896	0.815	1.277	0.902	0.932	0.857
UserKNN	0.960	0.707	0.935	0.864	0.836	0.634	0.952	0.883
Item-KNN	1.045	0.790	0.897	0.823	0.890	0.683	0.953	0.852
MF	0.925	0.686	**0.941**	**0.870**	**0.786**	**0.599**	**0.960**	**0.891**
NeuMF	0.943	0.694	0.936	0.866	0.819	0.634	0.955	0.886
PETER+	**0.924**	0.685	**0.941**	**0.870**	0.803	0.621	0.958	0.889
PEPLER-MLP	0.925	**0.672**	**0.941**	0.869	0.793	0.606	0.959	0.889
UPR (Llama2)	0.944	0.678	0.938	0.866	0.804	0.616	0.955	0.885
UPR (Mistral)	0.941	0.679	0.940	**0.870**	0.804	0.610	0.952	0.887

stream recommendation task. In particular, we prompt LLama2 to generate NL profiles but only use reviews from the top-k feature for each user, where $k \in \{1, 2, 3, 4, 5\}$. Each set of NL profiles is then used to fine-tune UPR.

Observing the changes in RMSE and MAE in Figure 2, a discernible trend emerges: augmenting the count of features reduces the loss, but the improvement in performance sees diminishing returns as additional features are incorporated. This observed trend is rationalized by the arrangement of features in order of significance, implying that subsequently added features hold lesser relevance for the user. Furthermore, the diminishing returns of adding features provides empirical evidence that the feature ranking methodology in Section 3.2 helps select the best features to use in the profile.

An important point to note is that the incorporation of an excessive number of lower-ranked features may potentially introduce noise into the user profile. Given that a user profile can only accommodate a finite amount of text, this noise might, in turn, compromise the efficacy of the recommendations provided. As a follow-up study, we plan to research how to strategically select the number of features and reviews that best capture a user's holistic preferences and optimize the recommendation performance. Furthermore, we set the max token limit of an NL profile to 200 tokens. We found that this was qualitatively a reasonable max length for a profile. However, more research is needed to study the tradeoffs between the length of the profile with regard to recommendation performance and cognitive load for users.

Table 3: User study measuring the quality of the generated NL profiles. We report the average number of satisfactory samples out of 50 samples per domain.

Metric	Amazon-MT	TripAdvisor
Fluency	95%	92%
Informativeness	86%	82%
Conciseness	75%	72%
Relevance	90%	87%

5.3 Qualitative Case Study on NL Profiles

As there is no ground-truth data available for the NL profile generation task, we instead assess the quality of our generated NL profiles by conducting a qualitative case study with users. Five participants, all of whom are master's students at a university, were shown 50 NL profiles from each dataset and were tasked with answering whether each NL profile met each of the following criteria:

1. **Fluency**: Is the NL profile both syntactically and semantically correct?
2. **Informativeness**: Does the NL profile provide important information for a user profile?
3. **Conciseness**: Is the NL profile written in a concise manner?
4. **Relevance**: Given the list of reviews, is the NL profile relevant to the user?

Participants were shown positive and negative examples per criteria in order to improve annotation quality. Each question is answered with a "yes" or "no". We show the results of the user study in Table

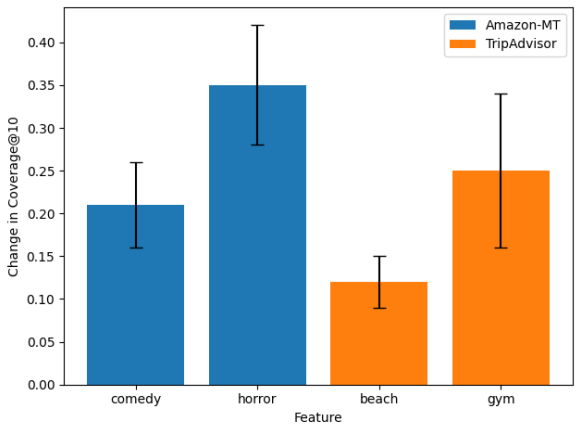

Figure 3: Change in Coverage@10 between edited profile versus original profiles when adding a new preference for a target feature.

3. The Fleiss' kappa coefficient among annotators was 0.82, suggesting a very high level of agreement. Overall, annotators were highly satisfied with the quality of our generated NL profiles. Both Amazon-MT and TripAdvisor scored well on fluency, informativeness, and relevance but lower on conciseness. During follow-up discussions, annotators felt that the profiles contained few grammatical errors and were generally relevant and informative. However, they also pointed out that profiles could be excessively long at times and contain redundant information. Given that we used a zero-shot approach to generate profiles, we anticipate that NL profiles can be improved through either additional prompt engineering or fine-tuning on the NL profile generation task.

From manual inspection of the NL profiles for Amazon-MT, we noticed that Llama2 and Mistral were sometimes able to infer actors and characters of a film based on the review, even when not explicitly mentioned. Although this capability can potentially improve the downstream recommendation task, the ability to infer information can also lead to hallucinations. Consequently, the model can also generate inaccurate facts and negatively impact performance. Further research is needed to ensure that the NL profile generated is factual and aligns with the user's interests.

5.4 Scrutinising NL Profile Preferences

To evaluate the scrutability capabilities of UPR, we simulate a user adding a new preference to their NL profile. In particular, we select 200 random users from the original profiles O that do not express any preference towards target feature t. We then prompt Mistral-Instruct-7B to edit the profile such that the user likes t (shown in Appendix A.5) to obtain the edited profile set C. We randomly sample 100 items from the test set where feature $f = t$ and 100 items where $f \neq t$. Finally, we run the model at inference time and calculate the Coverage@10, which is defined as the number of items in the top-10 recommendations where $f = t$, divided by 10. We repeat this sampling process five times on both datasets and use different seeds for robustness. We report the change in Coverage@10 between O and C in Figure 3, defined as $\overline{C_t} - \overline{O_t}$.

We observe that the Coverage@10 for all t increases across all samples when using C, meaning that a user can simply add a preference towards t to the NL profile, and the model will update the recommendations accordingly. In addition, the variance of the change in coverage shows that the recommendations can be correctly altered consistently.

Importantly, UPR does not require further fine-tuning when adding new preferences because the model learned how to interpret NL profiles effectively during the training phase. Furthermore, the features of each item were not explicitly passed to the model during fine-tuning or inference, meaning that the model has learned the features of each item from the NL profiles and its own pre-trained knowledge.

Overall, we show that *UPR* is able to learn the features of each item during training, meaning that a user can scrutinize and edit their profile to instantly receive updated recommendations. In contrast, there is no simple way to update user preferences in traditional collaborative filtering methods. Consequently, a user would need to drastically change their interaction history to align the recommendations with their current interests.

It is important to note that the reviews used to determine the features during the sampling process may be written by an arbitrary user A in Amazon-MT rather than the target user B. Thus, user A might consider the item a 'comedy' movie, but user B may not use the same terminology because preferences are subjective. Furthermore, user B still expresses preferences on other features. This means that they may highly rate non-target items in both O and C, which can lead to a decreased change in coverage.

6 Conclusion and Future Work

In this work, we propose a novel recommendation model that replaces uninterpretable user embeddings with transparent, natural language profiles that holistically describe a user's preferences. We show that instruction prompting can be used with large language models to create fluent, informative, and personalized user profiles based on user reviews. By encoding user preferences using natural language, we are able to develop a recommender system that is both transparent and scrutable. Experimental results show that our method has comparable performance to popular recommender models. In addition, we show that by adding new preferences to the NL profile, we can quickly provide updated recommendations. For future work, we plan to investigate multi-turn updates to NL profiles to simulate user updates over time and explore how scrutability can be used to adjust recommendations to be less biased.

7 Limitations

The performance of UPR is heavily dependent on the quality of the user reviews in the dataset, meaning that preferences that are not explicitly mentioned often in the review corpora will not be captured in the NL profile. Furthermore, the average rating in both datasets is 4.0/5.0, meaning that the vast majority of NL profiles contain *only* positive preferences. Thus, the model does not have many negative preferences to train on, making it difficult to edit a negative preferences to the profile (shown in Appendix A.6).

Another major limitation of using LLMs for recommendation is that they are notably slower than traditional recommender systems. Thus, users would need to wait a significantly longer amount of time to receive recommendations, which can lead to lower levels of satisfaction.

Finally, the model is bottlenecked by the information contained in the NL profile, meaning that the length of the profile and the number of features impacts recommendation performance. However, there is a concern that a verbose NL profile increases cognitive load for users. In terms of practical utility, making the NL profile too long and difficult to scrutinize negatively impacts usability for users. Future work can explore the tradeoffs between recommendation performance and human preferences.

8 Ethical Considerations

There are several ethical considerations to consider when using large language models for recommendation. Firstly, LLMs may show bias towards more popular items, which can lead to lesser-known items being ignored. Since recommender systems learn from user interactions, a bias amplification loop can be created that causes a greater discrepancy between popular and less-popular items (Chen et al., 2023). Furthermore, LLMs have also been criticized for unfairness with regards to underrepresented demographics, highlighting the importance of metrics that evaluate various social biases in LLMs (Salutari et al., 2023). Finally, hallucinations are a problematic issue in LLMs because the model may generate plausible, but incorrect information. In the case of recommender systems, the model might create an NL profile with incorrect information or recommend an item that does not exist (Li et al., 2023b). Consequently, safeguards must be added to ensure that any output from the language model is truthful and accurate.

References

Charu C. Aggarwal. 2016. *Recommender Systems - The Textbook*. Springer.

Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. 2019. 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 (SIGIR '19)*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

Toine Bogers. 2018. Tag-based recommendation. In Peter Brusilovsky and Daqing He, editors, *Social Information Access - Systems and Technologies*, volume 10100 of *Lecture Notes in Computer Science*, pages 441–479. Springer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Shuo Chang, F. Maxwell Harper, and Loren Terveen. 2015. Using groups of items for preference elicitation in recommender systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work Social Computing*, CSCW '15, page 1258–1269, New York, NY, USA. Association for Computing Machinery.

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Trans. Inf. Syst.*, 41(3).

Xu Chen, Changying Du, Xiuqiang He, and Jun Wang. 2020. Jit2r: A joint framework for item tagging and tag-based recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177.

Shijie Geng, Zuohui Fu, Yingqiang Ge, Lei Li, Gerard de Melo, and Yongfeng Zhang. 2022. Improving personalized explanation generation through visualization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 244–255, Dublin, Ireland. Association for Computational Linguistics.

F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4).

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *Proceedings of the 25th International Conference on World Wide Web*.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate neural template explanations for recommendation. In *CIKM*.

Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized transformer for explainable recommendation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4947–4957, Online. Association for Computational Linguistics.

Lei Li, Yongfeng Zhang, and Li Chen. 2023a. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems*, 41(4):1–26.

Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. 2023b. Large language models for generative recommendation: A survey and visionary discussions. *arXiv preprint arXiv:2309.01157*.

Sheshera Mysore, Mahmood Jasim, Andrew Mccallum, and Hamed Zamani. 2023a. Editable User Profiles for Controllable Text Recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 993–1003, Taipei Taiwan. ACM.

Sheshera Mysore, Andrew Mccallum, and Hamed Zamani. 2023b. Large Language Model Augmented Narrative Driven Recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 777–783, Singapore Singapore. ACM.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Filip Radlinski, Krisztian Balog, Fernando Diaz, Lucas Dixon, and Ben Wedin. 2022. On natural language user profiles for transparent and scrutable recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2863–2874, New York, NY, USA. Association for Computing Machinery.

Tetsuya Sakai. 2007. Alternatives to bpref. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 71–78, New York, NY, USA. Association for Computing Machinery.

Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1257–1264, Red Hook, NY, USA. Curran Associates Inc.

Flavia Salutari, Jerome Ramos, Hossein A Rahmani, Leonardo Linguaglossa, and Aldo Lipani. 2023. Quantifying the bias of transformer-based language models for african american english in masked language modeling. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 532–543. Springer Nature Switzerland Cham.

Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. 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*, pages 890–896.

Shilad Sen, F. Maxwell Harper, Adam LaPitz, and John Riedl. 2007. The quest for quality tags. In *Proceedings of the 2007 ACM International Conference on Supporting Group Work*, GROUP '07, page 361–370, New York, NY, USA. Association for Computing Machinery.

Mohit Sharma, F. Maxwell Harper, and George Karypis. 2019. Learning from sets of items in recommender systems. *ACM Transactions on Interactive Intelligent Systems*, 9(4):1–26.

Nava Tintarev and Judith Masthoff. 2015. *Explaining Recommendations: Design and Evaluation*, 2 edition, pages 353–382. Springer US.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Khanh Hiep Tran, Azin Ghazimatin, and Rishiraj Saha Roy. 2021. Counterfactual explanations for neural recommenders. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1627–1631, New York, NY, USA. Association for Computing Machinery.

Quoc-Tuan Truong, Aghiles Salah, and Hady Lauw. 2021. Multi-modal recommender systems: Hands-on exploration. In *Fifteenth ACM Conference on Recommender Systems*, pages 834–837.

Zhouhang Xie, Sameer Singh, Julian McAuley, and Bodhisattwa Prasad Majumder. 2022. Factual and informative review generation for explainable recommendation. *ArXiv*, abs/2209.12613.

Yongfeng Zhang and Xu Chen. 2020. Explainable recommendation: A survey and new perspectives. *Found. Trends Inf. Retr.*, 14(1):1–101.

Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review? boost phrase-level sentiment labeling with review-level sentiment classification. In *SIGIR*.

A Appendix

A.1 NL Profile Generation

To generate the profiles, we use the top 5 features calculated per user in Section 3.2. We then use 5 random reviews from each feature as input for the prompt, shown in Table 5. We set the max tokens generated to 200, temperature to 0.7, and seed to 0.

A.2 Recommendation Task

We encode and fine-tune our model using the following prompt:

"{**profile**} Based on my user profile, from a scale of 1 to 5 (1 being the lowest and 5 being the highest), i would give {**title**} a rating of"

where {**profile**} is the user profile and {**title**} is the target item. In order to improve performance, we scale ratings from 1-5 to 0-1 when updating the loss per batch. Ratings are scaled back to their original values during evaluation.

A.3 Implementation Details

For the recommendation task, we experiment with the number of features k set between 1 to 5. For hyperparameter tuning, we experiment with a learning rate of 1e-3, 3e-4, 1e-5, learning scheduler of linear and cosine, and batch sizes of 8, 16, and 32. In addition we train the model for 10 epochs, with early stopping after 3 epochs of no improvement.

A.4 Qualitative Test Study

The participants are master's students from the United Kingdom who volunteered to participate in the study. The study took about 30 minutes on average. All participants agreed to allow the data to reported for research purposes only. There is no identifiable or sensitive information recorded in the user study.

Before annotating the samples, the participants were provided with both positive and negative examples for each of the four questions to help them better understand the task. The participants were then shown every sample from the randomly selected pool of profiles and were asked to indicate whether the profile fulfilled each of the four criteria.

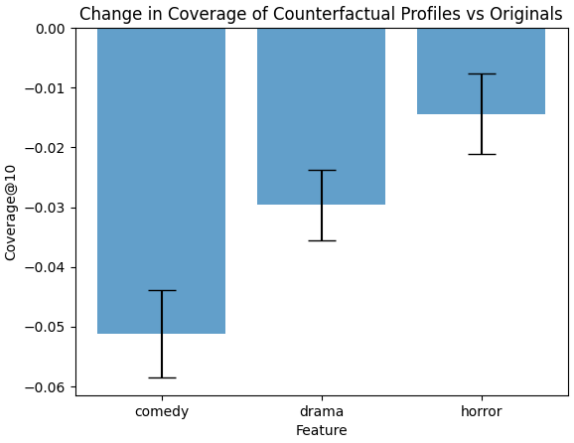

Figure 4: Change in Coverage@10 between counterfactual profile versus original profiles for a target feature.

A.5 Adding New Preferences to NL Profile

When sampling users and items from the test set, we use seeds=$\{0, 42, 100, 200, 300\}$. To generate the edited profile set C, we a few shot prompt, shown in Table 5, with max token set to 300, temperature of 0.7, and seed of 0.

During experimentation, we found that we needed to increase the max token size of the profiles to yield a change in recommendations because the profile needs more tokens to express a new preference in the NL profile. To obtain recommendations, we use the same prompt in Appendix A.2 using our fine-tuned model at inference time.

A.6 Editing Negative Preferences to NL Profile

We experiment with editing an existing positive preference towards target feature t in a NL profile to a negative preference to see if a user can receive less recommendations of t by scrutinizing the profile. We repeat the same sampling method in Section 5.4, but instead select 200 users that positively mention target feature t in their profile. We then use the following prompt with Llama2 to counterfactually edit the profile such that the user does not like t.

"Rewrite the user profile so that the user does not like beach movies. Keep the profile as similar as possible for all other preferences: {**profile**}."

To keep the counterfactual profiles as similar as possible to the original profiles, excluding t, we use the same hyperparameters in Appendix A.1.

We record the change in coverage for the top-10 recommendations between the counterfactual versus original profiles in Figure 4. We observe that the coverage drops, as expected, but the overall change is minor across the three features we tested. This can be attributed to the strong skew towards positive reviews in the data, meaning that vast majority of NL profiles used during training do not contain any negative preferences. In future work, we plan to train the model using more negative preferences in order to realize dislikes in the updated recommendations.

Table 4: We prompt an instruction-tuned LLM with the relevant reviews to generate cohesive and relevant NL profiles. The result is one NL profile included in our dataset.

Instruction: Summarize in a single paragraph using the first person my general movie and tv preferences based on my reviews. Do not mention the word reviews. **Reviews**: - not the most brilliant actress even in the best of circumstances - in the past 5 years or so i've watched close to 8 to 10 films about elizabeth i ... - bergman & fellini (tim burton in this century) remain my all-time favorite directors
Result: I enjoy watching movies and TV shows, particularly those with strong performances and well-crafted stories. I have a preference for classic films and directors such as Bergman and Fellini, and I appreciate a good adaptation of a play. I also enjoy watching actors take on complex roles and explore their craft. While I have mixed opinions on some actresses, I generally enjoy watching a talented performance. Recently, I have been drawn to films about Elizabeth I, and I appreciate a good historical drama. Additionally, I find it refreshing when a director can successfully weave together multiple storylines and timelines into a cohesive and engaging film.

Table 5: A few shot prompt example to add a new preference towards target feature t to the profile. For all few shot prompts used for this experiment, refer to our repository.

Modify the user profile so that the user also likes comedy movies. Keep all the profile as similar as possible for all other preferences.

Original Profile: I generally prefer action-packed movies and TV shows with plenty of stunts and excitement, but I also appreciate subtle moments of quiet subtlety and powerful performances. I prefer to watch DVDs instead of most TV shows, and I have a special interest in the works of certain directors like Steven Spielberg and Michael Mann. I have been impressed by the recent works of directors like Steven Soderbergh, and I believe they are making great strides in modern cinema.

New Profile: I generally prefer action-packed movies and TV shows with plenty of stunts and excitement, and now I've found a liking for comedy movies too, enjoying their humor alongside the subtle moments of quiet subtlety and powerful performances that I appreciate. I prefer to watch DVDs over TV shows, maintaining a special interest in directors like Steven Spielberg and Michael Mann. The recent works of directors like Steven Soderbergh also impress me, as I believe they contribute significantly to modern cinema.

Original Profile: As for my movie preferences, I tend to enjoy films that offer a unique perspective and don't rely too heavily on cliches. I appreciate when there is a clear focus on character development and backstory, but I also enjoy when filmmakers take risks and try something new. I have mixed feelings about movies that try to tackle complex issues like bioterrorism, as they can often feel overly sensationalized or heavy-handed. Ultimately, I'm drawn to movies that offer a fresh take on familiar themes and have a strong sense of style and pacing.

New Profile: As for my movie preferences, I tend to enjoy films that offer a unique perspective, including comedy movies, and steer clear of relying too heavily on cliches. I appreciate clear focus on character development and backstory, and value when filmmakers take risks and introduce humor alongside their innovation. My views on movies tackling complex issues like bioterrorism are mixed, as they can feel sensationalized. Ultimately, I'm drawn to movies with a fresh take on familiar themes, strong sense of style, pacing, and those that blend genres effectively, including smart comedies that offer insightful laughs.

Original Profile: I have a preference for movies and TV shows that showcase talented actors, particularly those who can bring depth and nuance to their roles. While I appreciate the performances of actors like John Malkovich and Danny DeVito, I sometimes wish for a more convincing fit in certain roles. However, I also enjoy a good story and will often watch movies and shows regardless of the actors involved. I have a tendency to be critical in my reviews, but I also believe in being honest and providing constructive feedback.

Result:

I have a preference for movies and TV shows that showcase talented actors, including those who excel in comedy, and appreciate depth and nuance in their roles. While I admire the performances of actors like John Malkovich and Danny DeVito, I sometimes wish for a more convincing fit in certain roles. However, I also enjoy a good story and will often watch movies and shows regardless of the actors involved. I strive to be honest and constructive in my reviews, providing valuable feedback for the benefit of both creators and fellow viewers.