Latent Linear Critiquing for Conversational Recommender Systems

Kai Luo
University of Toronto
kluo@mie.utoronto.ca

Scott Sanner*
University of Toronto
ssanner@mie.utoronto.ca

Ga Wu*
University of Toronto
wuga@mie.utoronto.ca

Hanze Li
University of Toronto
litos.li@mail.utoronto.ca

Hojin Yang
University of Toronto
hojin@mie.utoronto.ca

ABSTRACT

Critiquing is a method for conversational recommendation that iteratively adapts recommendations in response to user preference feedback. In this setting, a user is iteratively provided with an item recommendation and attribute description for that item; a user may either accept the recommendation, or critique the attributes in the item description to generate a new recommendation. Historical critiquing methods were largely based on explicit constraint- and utility-based methods for modifying recommendations w.r.t. critiqued item attributes. In this paper, we revisit the critiquing approach in the era of recommendation methods based on latent embeddings with subjective item descriptions (i.e., keyphrases from user reviews). Two critical research problems arise: (1) how to co-embed keyphrase critiques with user preference embeddings to update recommendations, and (2) how to modulate the strength of multi-step critiquing feedback, where critiques are not necessarily independent, nor of equal importance. To address (1), we build on an existing state-of-the-art linear embedding recommendation algorithm to align review-based keyphrase attributes with user preference embeddings. To address (2), we exploit the linear structure of the embeddings and recommendation prediction to formulate a linear program (LP) based optimization problem to determine optimal weights for incorporating critique feedback. We evaluate the proposed framework on two recommendation datasets containing user reviews with simulated users. Empirical results compared to a standard approach of averaging critique feedback show that our approach reduces the number of interactions required to find a satisfactory item and increases the overall success rate.

KEYWORDS

Conversational Recommendation, Critiquing

[*] Affiliate to Vector Institute of Artificial Intelligence, Toronto

WWW '20, April 20–24, 2020, Taipei, Taiwan
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380003

ACM Reference Format:
Kai Luo, Scott Sanner, Ga Wu, Hanze Li, and Hojin Yang. 2020. Latent Linear Critiquing for Conversational Recommender Systems. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3380003

1 INTRODUCTION

Critiquing is a method for conversational (a.k.a. sequential interactive) recommendation that adapts recommendations in response to user preference feedback regarding item attributes. For example, in unit critiquing [1], a user might critique a digital camera recommendation by requesting an item with higher resolution and in compound critiquing [10, 12], a user might further explore items that have longer battery life *and* lower price than an initial recommendation. Further extensions such as incremental critiquing consider the cumulative effect of iterated critiquing interactions [11] while experience-based methods attempt to collaboratively leverage critiquing interactions from multiple users [8].

Historical critiquing methods were largely based on constraint- and utility-based methods for modifying recommendations w.r.t. critiques of explicitly known item attributes. In this paper, we revisit the critiquing approach in the era of state-of-the-art recommendation methods based on latent embeddings [4, 5, 13, 14, 18]. In addition, we assume that item attributes are *not explicitly known*, but rather represented as keyphrases sourced from subjective user reviews. Though some work has focused on explanations in critiquing [12] and other work [2, 16, 17] has respectively explored speech- and dialog-based interfaces for critiquing-style frameworks, these architectures have nonetheless assumed that item attributes are largely orthogonal and explicitly known *a priori*, or otherwise could only handle a single critiquing step before resetting [17].

In our novel setting of *multistep latent critiquing*, it is not immediately clear how critiques of subjective keyphrases should be incorporated into the latent user preference representation to modulate future recommendations. To this end, we formalize and address two critical research problems in this paper:

(1) How can we co-embed item critiques with general user preference information to properly take the critiques into account for subsequent recommendations?
(2) How can we appropriately modulate the strength of each critique in multi-step critiquing feedback, where critiques are not necessarily independent, nor of equal importance?

To address problem (1), we extend an existing state-of-the-art linear embedding recommendation algorithm that we term Projected

Linear Recommendation (PLRec) [13] to embed user preferences and align review-based keyphrase attributes with those embeddings. To address problem (2), we exploit the linear structure of the embeddings and recommendation prediction to formulate a linear program (LP) based optimization problem to determine optimal weights for incorporating critique feedback. To this end, we refer to our overall framework as *latent linear critiquing* (LLC) since it exploits linear embedding structure to perform latent critiquing.

We evaluate LLC on two recommendation datasets containing user reviews with simulated users. Empirical results compared to standard approaches for averaging critique feedback show that our LP LLC approaches reduce the number of interactions required to find a satisfactory item and increases the overall percentage of successfully retrieved items in our experimental setting. In summary, this paper provides a novel critiquing method for manipulating latent user embeddings through efficient LP-based optimization.

2 PRELIMINARIES

2.1 Notation

Before proceeding, we define the following notation:

- $R \in \mathbb{R}^{|I| \times |J|}$. This is a user preference matrix. Entries $r_{i,j}$ are either 1 (preference observed) or 0 (preference not observed). $\mathbf{r}_i$ represents all feedback from user i, and $\mathbf{r}_{:,j}$ represents all user feedback for item j.
- $S \in \mathbb{R}^{|I| \times |K|}$. This is the user-keyphrase matrix. Given user reviews from a corpus, we extract keyphrases that describe item attributes from all reviews as shown in Table 1. This matrix contains user and term frequencies of keyphrases. We use $\mathbf{s}_i$ for ith user's count of keyphrase usage in their reviews, and $\mathbf{s}_{:,k}$ for the kth keyphrase count over all users.
- $S' \in \mathbb{R}^{|J| \times |K|}$. This is the item-keyphrase matrix. Given item reviews written by all users from a corpus, we extract keyphrases that describe item attributes as shown in Table 1. This matrix contains total keyphrase counts (frequencies) in reviews for each item and keyphrase term. We use $\mathbf{s}'_j$ to represent jth item's keyphrase frequencies, and $\mathbf{s}'_{:,k}$ to represent kth keyphrase's frequency across all items.
- $j^{-k} \in \{j | S'_{j,k} = 0, \forall j\}$. This item set represents items that do not contain the critiqued keyphrase k.
- $j^{+k} \in \{j | S'_{j,k} > 0, \forall j\}$. This item set represents items that contain the critiqued keyphrase k.

2.2 Projected Linear Recommendation

Linear Recommendation methods learn a matrix representing either user-user or item-item similarity by casting the problem in a linear regression framework [9, 15]. An unconstrained linear recommender uses an objective of the following form:

$$\underset{W}{\operatorname{argmin}} \sum_i \left\| \mathbf{r}_i - \mathbf{r}_i W^T \right\|_2^2 + \Omega(W), \quad (1)$$

where W represents the similarity matrix to train, and Ω is a regularization term. Unfortunately, in modern recommendation problems that have large numbers of users and items, such naïve linear recommendation approaches are infeasible due to the space requirements of explicitly storing the similarity matrix W.

Table 1: The two datasets we use in this paper along with example keyphrases extracted from reviews in the dataset.

Dataset	Reason Type	Keyphrases
Beer	Head	white, tan, offwhite, brown
	Malt	roasted, caramel, pale, wheat, rye
	Color	golden, copper, orange, black, yellow
	Taste	citrus, fruit, chocolate, cherry, plum
CDs&Vinyl	Genre	rock, pop, jazz, rap, hip hop, R&B
	Instrument	orchestra, drum
	Style	concert, opera
	Religious	chorus, christian, gospel

Projected Linear Recommendation (PLRec) is a term that we use for the state-of-the-art recommendation method of [13] that mitigates the scalability problem described above by projecting preferences from R into a reduced-dimension embedded space prior to linear regression. Formally, the PLRec objective is defined as

$$\underset{W}{\operatorname{argmin}} \sum_i \left\| \mathbf{r}_i - \mathbf{r}_i V W^T \right\|_2^2 + \Omega(W), \quad (2)$$

where parameters W are learned and V is a fixed embedding projection matrix. It is critical to note here that because V is fixed, the above objective still leads to a convex linear regression problem. PLRec obtains V by taking a low-rank SVD approximation of the observation matrix R such that $R = U\Sigma V^T$, and the rank $|L|$ of V is far smaller than the observation dimensions $|I|$ and $|J|$. We denote the projected, embedded representation of user i as $\mathbf{z}_i = \mathbf{r}_i V$.

2.3 Conversational Critiquing

In the conversational critiquing setting of this paper, a user is iteratively provided with item recommendations and keyphrase descriptions for that item; a user may either critique the keyphrases in the item description or accept the item recommendation, at which point the iteration terminates. Two examples of a conversational critiquing interaction from our experiments are provided in Table 2.

A single critiquing step can be viewed as a series of functional transformations that produce a modified prediction $\hat{\mathbf{r}}_i$ of item preferences for user i given critiqued item keyphrases $\mathbf{s}_i$ as follows:

$$\hat{\mathbf{r}}_i = f_m(\mathbf{r}_i, \tilde{\mathbf{s}}_i), \quad \textit{given} \quad \tilde{\mathbf{s}}_i = \psi(\mathbf{s}_i, \mathbf{c}_i), \quad (3)$$

where the critique-modified recommendation function f_m takes user preferences $\mathbf{r}_i$ and critiqued keyphrases $\tilde{s_i}$ as input and produces a recommendation $\hat{\mathbf{r}}_i$ as output. The function ψ applies a user critiquing action $\mathbf{c}_i$ to user keyphrases $\mathbf{s}_i$.

In the case of critiquing over multiple time steps as demonstrated in Figure 1, a user is iteratively provided with the item recommendations $\tilde{\mathbf{r}}_i^t$ for each time step t, and, based on the recommendations, the user may make a new critique $\mathbf{c}_i^t$ and update the representation of critiqued keyphrases $\tilde{\mathbf{s}}_i^{t-1}$ though a cumulative critiquing function ψ:

$$\tilde{\mathbf{s}}_i^t = \psi(\mathbf{s}_i, \tilde{\mathbf{s}}_i^{t-1}, \mathbf{c}_i^t). \quad (4)$$

The user may alternately accept the item recommendation, at which point the iteration terminates.

Table 2: User Case Study for target rank 1 on the CDs&Vinyl and Beer datasets.

Dataset	Time Step t	Recommended Item	Top Keyphrases Describing the Item	Critiqued Keyphrase	Successfully Retrieved?
CDs&Vinyl	0	No Code	Rock, Metal, Rap	Rap	-
	1	Jar Of Flies	Rock, Pop, Ballad	Pop	-
	2	Pearl Jam	Rock, Metal, Instrument	-	✓
Beer	0	Sierra Nevada Southern Hemisphere Harvest Fresh Hop Ale	Citrus, Sweet, Caramel	Citrus	-
	1	Maharaja	Sweet, Orange, Grapefruit	Grapefruit	-
	2	Fuller's ESB	Caramel, Fruit, Copper	-	✓

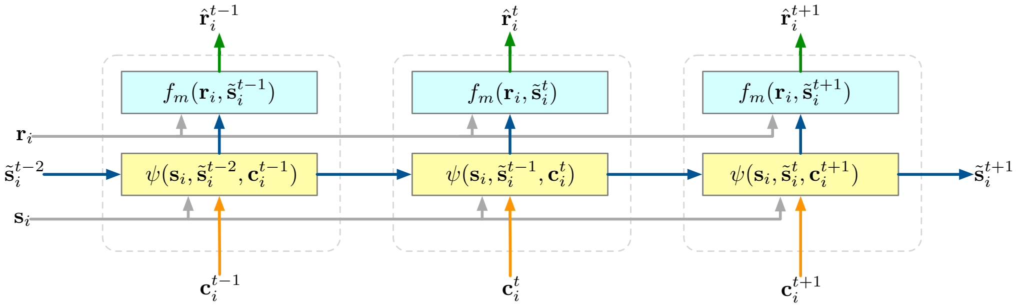

$\hat{\mathbf{r}}_i^{t-1}$
$\hat{\mathbf{r}}_i^{t}$
$\hat{\mathbf{r}}_i^{t+1}$
$f_m(\mathbf{r}_i, \tilde{\mathbf{s}}_i^{t-1})$
$f_m(\mathbf{r}_i, \tilde{\mathbf{s}}_i^{t})$
$f_m(\mathbf{r}_i, \tilde{\mathbf{s}}_i^{t+1})$
$\mathbf{r}_i$
$\tilde{\mathbf{s}}_i^{t-2}$
$\psi(\mathbf{s}_i, \tilde{\mathbf{s}}_i^{t-2}, \mathbf{c}_i^{t-1})$
$\psi(\mathbf{s}_i, \tilde{\mathbf{s}}_i^{t-1}, \mathbf{c}_i^{t})$
$\psi(\mathbf{s}_i, \tilde{\mathbf{s}}_i^{t}, \mathbf{c}_i^{t+1})$
$\tilde{\mathbf{s}}_i^{t+1}$
$\mathbf{s}_i$
$\mathbf{c}_i^{t-1}$
$\mathbf{c}_i^{t}$
$\mathbf{c}_i^{t+1}$

Figure 1: The flow of conversational critiquing over three time steps. Previously critiqued keyphrases $\tilde{\mathbf{s}}_i^{t-2}$ at time $t-2$ are combined by ψ with the newly critiqued keyphrase $\mathbf{c}_i^{t-1}$ at time $t-1$ to yield $\tilde{\mathbf{s}}_i^{t-1}$. The recommendation produced by f_m at time $t-1$ for user i is produced from the user's historical preferences $\mathbf{r}_i$ and cumulative critiques $\tilde{\mathbf{s}}_i^{t-1}$ up to time $t-1$. This process repeats for all t until the user accepts the recommendation (or terminates) and ceases to provide additional critiques.

At this point we have defined the generic functional structure of the overall critiquing process, but we still need to provide formal definitions of f_m and ψ to address the critical problem of how keyphrase critiques modulate recommendations. We now proceed to do this in the specific setting of latent linear conversational critiquing that extends and leverages the linearity of PLRec.

3 LATENT LINEAR CONVERSATIONAL CRITIQUING

In this section, we begin by extending the PLRec framework to embed language-based feedback mined from user reviews that will be used for co-embedding critiques with user preferences from PLRec. We then show how to combine critiques in a linear framework where we propose methods for both simple critique averaging and a more sophisticated framework for deriving critique weights by leveraging a novel linear programming optimization formalization.

3.1 Co-embedding of Language-based Feedback

First we augment the PLRec framework with the ability to embed language-based critiquing feedback in the same space as user preferences. To achieve this, we leverage recommendation datasets with both preference and review feedback from users.

PLRec naturally embeds user preferences over items into a latent space as described in Equation (2). Once this latent embedding for a user is obtained, we learn to embed keyphrase-based critiques for a user by training to recover a user's latent preference embedding *from* the preference content implicitly revealed via their reviews.

To make this concrete, for each user i, PLRec encodes their latent preference representation $\mathbf{z}_i$ as in Equation (2). In addition, we have review content for each user represented as a term frequency vector (user keyphrases) $\mathbf{s}_i$. With this, we can now cast the co-embedding task as the following linear regression problem:

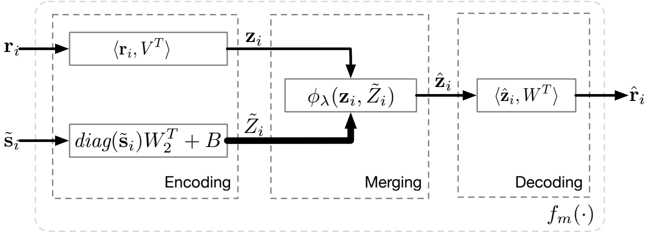

$\mathbf{r}_i$
$\langle \mathbf{r}_i, V^T \rangle$
$\mathbf{z}_i$
$\tilde{\mathbf{s}}_i$
$diag(\tilde{\mathbf{s}}_i)W_2^T + B$
$\tilde{Z}_i$
Encoding
$\phi_\lambda(\mathbf{z}_i, \tilde{Z}_i)$
$\hat{\mathbf{z}}_i$
Merging
$\langle \hat{\mathbf{z}}_i, W^T \rangle$
$\hat{\mathbf{r}}_i$
Decoding
$f_m(\cdot)$

Figure 2: The structure of critiqued recommendation $f_m(\cdot)$. Historical user preferences $\mathbf{r}_i$ and critiqued keyphrases $\tilde{\mathbf{s}}_i$ from user i flow in at the left and are embedded into respective latent spaces (i.e., a single vector embedding $\mathbf{z}_i$ for preferences and a matrix $\tilde{Z}_i$ containing embeddings for each critique). These latent embeddings are then combined in the merging stage (a critical contribution of the paper) and finally decoded into new item recommendations $\hat{\mathbf{r}}_i$ for user i on the right that take into account their critiques.

$$\underset{W_2,\mathbf{b}}{\text{argmin}} \sum_i \|\mathbf{z}_i - \tilde{\mathbf{z}}_i\|_2^2 + \Omega(W_2), \quad (5)$$

where

$$\tilde{\mathbf{z}}_i = \mathbf{s}_i W_2^T + \mathbf{b} \quad (6)$$

and $W_2 \in \mathbb{R}^{|K|\times|L|}$ projects users' review text to into their latent representation and $\mathbf{b}$ is a bias term. We will show how to use the learned regression model to conduct critiquing in next section.

3.2 Latent Linear Critiquing

In this section, we aim to specify how the critiquing-based recommendation system can make new recommendations after a user i has provided critiques $\mathbf{c}_i^1 \cdots \mathbf{c}_i^t$ over t iterations (as demonstrated in Figure 1 and 2), where critiques $\mathbf{c}_i^t$ are encoded as one-hot keyphrase indicators that represent a user i's dislike of a keyphrase description at time step t.

We start off by recalling that for a fixed user i and item j, the prediction of an item preference $\hat{r}_{i,j}$ can be written as an inner product $\langle \cdot, \cdot \rangle$ of the user and item embedding:

$$\hat{r}_{i,j} = \langle \mathbf{z}_i, \mathbf{w}_j \rangle , \tag{7}$$

where user representation $\mathbf{z}_i$ comes from the embedding of a user's historical preferences and $\mathbf{w}_j$ is the row of W corresponding to item j's latent embedding.

As mentioned previously, to transform the user's critiques $\mathbf{c}_i$ into an embeddable term frequency representation that can be co-embedded with user preference embeddings, we use the cumulative critiquing function defined as follows:

$$\tilde{\mathbf{s}}_i^t = \psi(\mathbf{s}_i, \tilde{\mathbf{s}}_i^{t-1}, \mathbf{c}_i^t) = \tilde{\mathbf{s}}_i^{t-1} - \max(\mathbf{s}_i, 1) \odot \mathbf{c}_i^t , \tag{8}$$

where $\odot$ represents element-wise multiplication of two vectors and the initial $\tilde{\mathbf{s}}_i^0$ is a zero vector of length $|K|$. The rationale for using $\max(\mathbf{s}_i, 1) \odot \mathbf{c}_i^t$ is twofold: (i) the critiqued keyphrase should be embedded with a strength similar to the user's overall frequency usage of the keyphrase (otherwise the linear embedding may have a small magnitude), hence $\mathbf{s}_i \odot \mathbf{c}_i^t$, but (ii) if the user did not use the keyphrase, it should receive a non-zero frequency, hence $\max(\cdot, 1)$.

With the critiqued keyphrases $\tilde{\mathbf{s}}_i^t$ and the mapping between keyphrase and user latent representation learned in Equation (5), we provide a latent representation of *all* critiques in matrix form

$$\tilde{Z}_i^t = diag(\tilde{\mathbf{s}}_i^t) W_2^T + B, \tag{9}$$

where each row $\tilde{\mathbf{z}}_i^k$ of the matrix $\tilde{Z}_i$ represents latent representation of the kth critiqued keyphrase, and each row of B is the identical bias term $\mathbf{b}$. While the matrix $\tilde{Z}_i$ appears large, it is sparse and simply used here for notational convenience to represent *stacking* of all critiqued keyphrase embeddings in matrix form.

Under an assumption that critiques at each time step should be weighted equally (with the user preference), we define simple **Uniform Average Critiquing** by specifying the merging function

$$\phi_\lambda(\mathbf{z}_i, \tilde{Z}_i^t) = \lambda_0 \mathbf{z}_i + \lambda_1 \tilde{\mathbf{z}}_i^1 \cdots \lambda_{|K|} \tilde{\mathbf{z}}_i^{|K|}, \tag{10}$$

where $\lambda_0, \lambda_1, \ldots, \lambda_{|K|}$ are identical and sum to 1 (hence all λ's equal $\frac{1}{|K|+1}$). Note that we omit the identical time step t superscripts to reduce notational clutter. Clearly, this merging function averages the user preferences and critiqued keyphrase embeddings uniformly. We can then obtain the *refined* recommendation score $\hat{r}_{i,j}^t$ of each user i for item j at time step t by decoding the latent representation and rearranging terms as follows:

$$\begin{aligned} \hat{r}_{i,j}^t &= \langle \phi_\lambda(\mathbf{z}_i, \tilde{Z}_i), \mathbf{w}_j \rangle \\ &= \langle (\lambda_0 \mathbf{z}_i + \lambda_1 \tilde{\mathbf{z}}_i^1 + \cdots + \lambda_{|K|} \tilde{\mathbf{z}}_i^{|K|}), \mathbf{w}_j \rangle \\ &= \frac{1}{|K|+1} \left(\langle \mathbf{z}_i, \mathbf{w}_j \rangle + \langle \tilde{\mathbf{z}}_i^1, \mathbf{w}_j \rangle + \cdots + \langle \tilde{\mathbf{z}}_i^{|K|}, \mathbf{w}_j \rangle \right) , \end{aligned} \tag{11}$$

where the second line substitutes the definition of $\phi_\lambda(\mathbf{z}_i, \tilde{Z}_i)$ from Equation (10) and the final line distributes the inner product over the summation but factors out the common coefficient.

An alternative approach we call **Balanced Average Critiquing** averages critique embeddings $\tilde{Z}_i$ together and then averages them again with the user preference embedding, thereby balancing the critique and user embeddings. This user-balanced average reflects the notion that user preferences after critiques should not deviate too drastically from the base user preferences to begin with. Formally, we define the merging function as

$$\begin{aligned} \phi_\lambda(\mathbf{z}_i, \tilde{Z}_i^t) &= \lambda_0 \mathbf{z}_i + \lambda_1 \tilde{\mathbf{z}}_i^1 + \cdots \lambda_{|K|} \tilde{\mathbf{z}}_i^{|K|} \\ &= \frac{1}{2} \left(\mathbf{z}_i + \frac{1}{|K|} \left(\tilde{\mathbf{z}}_i^1 + \cdots \tilde{\mathbf{z}}_i^{|K|} \right) \right), \end{aligned} \tag{12}$$

where $\lambda_0 = \frac{1}{2}$ and the remaining λ's equal $\frac{1}{2 \times |K|}$.

Unfortunately, the simple average critique weighting assumption described above can be violated in a few ways in practice:

- Not all critiques are equal. More specifically, later critiques may be more subtle refinements of earlier critiques.
- Not all critiques are independent. For instance, two critiques may substantially overlap while another critique may be more orthogonal to the first two and deserve more weight.

Therefore, we look for better solutions by seeking a way to optimize the weights used to merge critiques.

3.3 LP-based Critiquing Optimization

Based on the above criticisms, we revisit the merging function in the linear form of Equation (10) for arbitrary λ's and consider that each critique embedding has an additive linear impact on the rating:

$$\begin{aligned} r_{i,j}^t &= \langle \lambda_0 \mathbf{z}_i + \lambda_1 \tilde{\mathbf{z}}_i^1 \cdots \lambda_{|K|} \tilde{\mathbf{z}}_i^{|K|}, \mathbf{w}_j \rangle \\ &= \langle \lambda_0 \mathbf{z}_i, \mathbf{w}_j \rangle + \langle \lambda_1 \tilde{\mathbf{z}}_i^1, \mathbf{w}_j \rangle + \cdots + \langle \lambda_{|K|} \tilde{\mathbf{z}}_i^{|K|}, \mathbf{w}_j \rangle . \end{aligned} \tag{13}$$

Now, the question is how can we automatically find a better non-uniform weighting, i.e., optimal λ values?

In short, we can view this question as a formal optimization of the λ's in Equation (13). While the design of such an optimization problem is inherently subjective, there are a few key desiderata:

- The rating of items *known* to have the critiqued keyphrase description (j^{+k}) should be minimized.
- Inversely, the rating of items *not known* to contain the critiqued keyphrase (j^{-k}) should be maximized.
- Because we are ranking and λ weights could be infinite if unbounded, we need to ensure the λ are finitely bounded.

3.4 Optimization Objective

Based on the above desiderata, we arrive at the following linear objective of optimal critique weighting for each time step t

$$\max_{\lambda_0, \ldots, \lambda_{|K|}} \sum_{j^{+k}} \sum_{j^{-k}} \left(\hat{r}_{i,j^{-k}}^t - \hat{r}_{i,j^{+k}}^t \right) , \tag{14}$$

where we choose user and critique embedding weights $\lambda_0, \ldots, \lambda_{|K|}$ to maximize the pairwise difference of ratings of non-critiqued items j^{-k} with critiqued items j^{+k}. To limit computational complexity, we consider the top-100 rated items meeting the criteria for j^{+k} and j^{-k} since other items are less likely to be seen by the user.

Critically, since all of the embeddings are fixed, we see that the objective in Equation (14) is linear in the optimized variables and this is in fact a simple linear programming (LP) formulation of our desiderata. This is a fortuitous observation since LPs can be solved efficiently at scale (millions of variables and constraints), which enables practical deployment of this optimized weighting approach.

We propose three different LP constraint candidates:

- **LP Option 1:** Here, we assume the weighting of the original user preferences remain unchanged ($\lambda_0 = 1$). For the critique weights λ_i ($i > 0$), we allow for the fact that some critiques may partially cancel out others (i.e., having a negative weight) in the event of non-independence:

$$\lambda_0 = 1, \lambda_{1 \cdots |K|} \in [-1, +1] \tag{15}$$

- **LP Option 2:** While we previously chose the critique embedding with the intent that a latent weighting of $\lambda_{1 \cdots |K|} \in [-1, +1]$ should be a sufficient range (cf. discussion following Equation (8)), we now check this assumption by simply increasing the allowable range from $[-1, 1]$ to $[-r, +r]$:

$$\lambda_0 = 1, \lambda_{1 \cdots |K|} \in [-r, +r], \quad \text{where} \quad r \in [1, \infty) \tag{16}$$

- **LP Option 3:** Here, we require all λ's to be positive and sum to one, forcing the λ's to "compete" with each other. We also allow the LP to zero out λ_0 to ignore base user preferences:

$$\sum_{n=0}^{|K|} \lambda_n = 1, \quad \textit{and} \quad \lambda_{0 \cdots |K|} \in [0, 1] \tag{17}$$

We remark that LP1 and LP2 with pure bound constraints admit a greedy solution since the optimal λ_i's must be at the bound limits. Though not intuitive, these extreme weights work well empirically.

4 EXPERIMENTS

4.1 Dataset and Code

We evaluate the LLC framework on two publicly available datasets using simulated users for large-scale comparative evaluation: BeerAdvocate [6] and Amazon CDs&Vinyl [3, 7]. Each of the datasets contain 100,000+ reviews and product rating records. For Top-N ranking evaluation, we binarize the rating column of both datasets with rating threshold ϑ ($\vartheta > 3$ out of 5 for CDs&Vinyl, $\vartheta > 4$ out of 5 for BeerAdvocate). Table 3 shows overall dataset statistics for our experiments. Code to reproduce all experiments is on github.[1]

4.2 Automated Keyphrase Extraction

We extract descriptive keyphrases from reviews (cf. Table 1) that may represent variant expressions for the same concept; this is different from aspects that tend to be independent attributes [19]. We used the following steps to extract candidate keyphrases:

1. Extract separate unigram and bigram lists of high frequency noun and adjective phrases from reviews of the entire dataset.
2. Prune the bigram keyphrase list using a Pointwise Mutual Information (PMI) threshold to ensure bigrams are statistically unlikely to have occurred at random.
3. Represent each review as a sparse 0-1 vector indicating whether each keyphrase occurred in the review.

Table 3: Summary of datasets. We selected 40 keyphrases for CDs&Vinyl and 75 keyphrases for BeerAdvocate. Keyphrase coverage shows the percentage of reviews/comments that have at least one selected keyphrase.

Dataset	# Users	# Items	$\|r_{i,j} > \vartheta\|$	Rating Sparsity	Keyphrase Coverage	Keyphrase Average Counts (per User)
CDs&Vinyl (Amazon)	6,056	4,395	152,670	0.5736%	75.48%	13.9969
Beer (BeerAdvocate)	6,370	3,668	263,278	1.1268%	99.29%	55.1088

Algorithm 1 User Simulation Evaluation

```
procedure Eval($R^{te}$ for test)
    for each user $i$ do
        for each target item $j$, where $r_{i,j}^{te} = 1$ do
            for time step $t \in range(1, MAX)$ do
                user act critique $\mathbf{c}^t$
                optimize $\lambda_0 ... \lambda_t$
                $\hat{r}_{i,j}^t \leftarrow \langle \lambda_0 \mathbf{z}_i + \lambda_1 \tilde{\mathbf{z}}_i^1 + \cdots \lambda_{|K|} \tilde{\mathbf{z}}_i^{|K|}, \mathbf{w}_j \rangle$
                if $j$ in Top-N recommendation list then
                    break session with success
            length $\leftarrow min(t, MAX)$
    return average success rate & length
```

4.3 Critiquing User Simulation Evaluation

In order to perform an evaluation of each model's performance in a multi-step conversational recommendation scenario using offline data provided in our datasets, we conduct an evaluation by user simulation. Concretely, as described in Algorithm 1, we track the conversational interaction session of simulated users by randomly selecting a *target item* from their test set, having the user critique keyphrases inconsistent with the target item, and repeating until an iteration limit or the target item appears within the top-N recommendations on that iteration. For each user, we collect results from three simulated sessions and estimate the average success rate as well as the average session length for model comparison. Two example user critiquing simulations with LP1 are shown in Table 2.

The Top-N ranking threshold in this experiment is selected from $\{1, 5, 10\}$, where success becomes easier with increasing N. The maximum allowed critiquing iterations in Algorithm 1 is set to 10.

4.4 Candidate Critiquing Algorithms

Before listing the critiquing algorithms we compare, we first introduce a method that has omniscient knowledge of target items and the correctness of keyphrase descriptions that we term "Oracular Upper Bound". In general we cannot assume keyphrase labels are accurate descriptions nor can we assume that all relevant keyphrase labels occur for an item (they are derived from sparse user review content) and hence in practice we cannot simply prune all items that contain a critiqued keyphrase. However, if we "cheat" and assume knowledge of the user simulation and the perfect nature of keyphrase labels that are provided to the user in the simulation, we can define the "Oracular Upper Bound" that perfectly prunes the recommendation list after each critique. Because this assumes

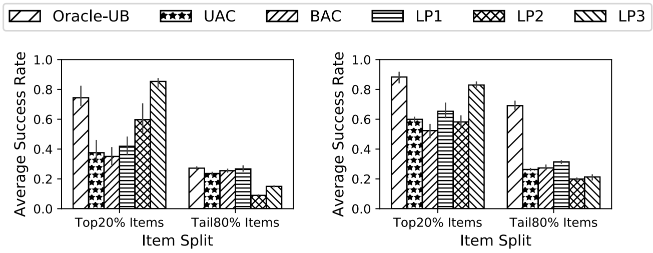
[1] https://github.com/k9luo/LatentLinearCritiquingforConvRecSys.

Table 4: Average percentage of target items successfully retrieved within Top-N (1,5,10) with 95% confidence intervals.

Model	CDs&Vinyl			Beer		
	Success Rate@1	Success Rate@5	Success Rate@10	Success Rate@1	Success Rate@5	Success Rate@10
Oracle-UB	0.1332±0.0037	0.2819±0.0085	0.3718±0.0033	0.4419±0.0199	0.7064±0.0258	0.7853±0.0141
UAC	0.1192±0.0066	0.2408±0.0115	0.3123±0.0091	0.12±0.0038	0.2847±0.0047	0.3932±0.016
BAC	0.1232±0.0063	0.2587±0.0075	0.3382±0.0105	0.1154±0.0047	0.2924±0.0139	0.3906±0.0164
LP1	**0.1301±0.0111**	**0.2722±0.0154**	**0.3426±0.0125**	**0.149±0.0072**	**0.3391±0.009**	**0.4524±0.006**
LP2	0.0379±0.0061	0.1025±0.0041	0.153±0.0044	0.0851±0.0066	0.2229±0.0086	0.3145±0.0107
LP3	0.0663±0.0059	0.1693±0.0035	0.2282±0.0071	0.0961±0.0125	0.2513±0.0175	0.3495±0.0024

Table 5: Average length of sessions to achieve Top-N (1,5,10) target ranks with 95% confidence intervals.

Model	CDs&Vinyl			Beer		
	Average Length@1	Average Length@5	Average Length@10	Average Length@1	Average Length@5	Average Length@10
Oracle-UB	4.5791±0.0665	3.9914±0.0486	3.6254±0.0525	5.1709±0.0588	3.8794±0.1312	3.3377±0.1227
UAC	9.0335±0.0384	7.9361±0.1104	7.2785±0.0841	6.1647±0.081	5.388±0.0765	4.8398±0.0166
BAC	8.984±0.0534	7.7693±0.0584	7.0648±0.089	6.3795±0.1538	5.535±0.141	5.0138±0.1103
LP1	**8.4136±0.0605**	**7.2963±0.1347**	**6.693±0.1276**	**5.7623±0.1357**	**4.9893±0.128**	**4.4925±0.1026**
LP2	9.674±0.0656	9.1579±0.0504	8.7434±0.0335	6.4092±0.0396	5.8067±0.056	5.3961±0.1076
LP3	9.4715±0.0417	8.629±0.0327	8.1493±0.078	6.4367±0.1385	5.7517±0.1646	5.3186±0.0679



(a) CDs&Vinyl

(b) Beer

Figure 3: Long tail item analysis average success rate at target rank 5 comparison for different critiquing algorithms on CDs&Vinyl and Beer with 95% confidence interval.

knowledge of the user simulation that actual critiquing methods cannot assume, this method serves as an approximate upper bound.

Now we list all critiquing methods used in our experiments:

- **Oracle-UB:** Oracular upper bound as defined above.
- **UAC:** Uniform Average Critiquing from Equation (10).
- **BAC:** Balanced Average Critiquing from Equation (12).
- **LP 1-3:** See respective Equations (15), (16) ($r = 100$), & (17).

4.5 Critiquing Empirical Results

4.5.1 Average Success Rate and Session Length. Table 4 shows the percentage of target items that successfully reach a rank of N ∈ {1, 5, 10} before the session terminates and Table 5 shows the average length of these sessions (both results averaged over users). From Table 3, we observe that the Beer dataset is much more dense than CDs&Vinyl in terms of the number of ratings per user and the number of keyphrases they used; this leads to marginally better performance of all methods on Beer than CDs&Vinyl since overall learning quality generally improves with information density.

The results illustrate that Oracle-UB behaves as an upper bound as expected. For the non-oracular methods, LP1 consistently outperforms the two baselines (UAC and BAC) and other LP variants across datasets and target ranks. LP1 finds a good weight trade-off between base user preferences and critique. LP3 shows a higher success rate than LP2 across datasets and target ranks. The average critiquing session length results closely reflect the success rate results — a higher success rate leads to earlier session termination.

Overall, we see that simply averaging the latent user and critique embeddings in UAC and BAC provide a reasonably strong baseline method, though there is no clear winner among the two across both datasets. We see that LP1 outperforms all other methods and further that LP2 and LP3 are weaker than LP1, UAC, and BAC, indicating that the nuanced choice of optimization objective and constraints is critical for good performance. The results suggest that the more extreme λ weight range of LP2 hurt performance and thus the restricted weight range of LP1 serves as a sort of "regularization" effect on how well the optimized weighting generalizes to the next step. Constraining total weight magnitude to force trade-offs in the weighting of each critique in LP3 appears to hurt performance.

4.5.2 Long-tail Item Analysis. In Figure 3, we split items by popularity into the top 20% tail 80% of items and report average success rate at target rank N = 5. All models perform better for the top 20% of items since they are more popular and likely to be recommended. LP3 provides strong performance on the top 20% items (outperforming even Oracle-UB on CDs&Vinyl) since it is the one method that can ignore the user embedding and rely solely on popularity bias. For the tail 80% of items, LP1 performs competitively showing that optimization combined with consideration of the base user embedding is helpful to retrieve more personalized long-tail items.

REFERENCES

[1] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. 1996. Knowledge-based Navigation of Complex Information Spaces. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1 (AAAI'96)*. AAAI Press, 462–468.

[2] Peter Grasch, Alexander Felfernig, and Florian Reinfrank. 2013. ReComment: Towards Critiquing-based Recommendation with Speech Interaction. In *Proceedings of the 7th ACM Conference on Recommender Systems (RECSYS)-13*. New York, NY, USA, 157–164.

[3] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 507–517.

[4] 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*. International World Wide Web Conferences Steering Committee, 173–182.

[5] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. Republic and Canton of Geneva, Switzerland, 689–698.

[6] Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 1020–1025.

[7] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.

[8] Kevin McCarthy, Yasser Salem, and Barry Smyth. 2010. Experience-based critiquing: Reusing critiquing experiences to improve conversational recommendation. In *International Conference on Case-Based Reasoning*. Springer, 480–494.

[9] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 497–506.

[10] James Reilly, Kevin McCarthy, Lorraine Mcginty, and Barry Smyth. 2004. Dynamic Critiquing. In *Advances in Case-Based Reasoning, 7th European Conference (ECCBR) 2004*. 37–50. https://doi.org/10.1007/978-3-540-28631-8_55

[11] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. 2004. Incremental critiquing. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 101–114.

[12] James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. 2005. Explaining Compound Critiques. *Artif. Intell. Rev.* 24, 2 (Oct. 2005), 199–220.

[13] Suvash Sedhain, Hung Bui, Jaya Kawale, Nikos Vlassis, Branislav Kveton, Aditya Krishna Menon, Trung Bui, and Scott Sanner. 2016. Practical linear models for large-scale one-class collaborative filtering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 3854–3860.

[14] S. Sedhain, A. Menon, S. Sanner, and L. Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on the World Wide Web (WWW-15)*. Florence, Italy.

[15] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Darius Braziunas. 2016. On the effectiveness of linear models for one-class collaborative filtering. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[16] Cynthia A Thompson, Mehmet H Goker, and Pat Langley. 2004. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research* 21 (2004), 393–428.

[17] Ga Wu, Kai Luo, Scott Sanner, and Harold Soh. 2019. Deep Language-based Critiquing for Recommender Systems. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys-19)*. Copenhagen, Denmark.

[18] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[19] Lei Zhang and Bing Liu. 2017. *Sentiment Analysis and Opinion Mining*. Springer US, Boston, MA, 1152–1161.