A Ranking Optimization Approach to Latent Linear Critiquing for Conversational Recommender Systems

HANZE LI, Mechanical and Industrial Engineering, University of Toronto, Canada
SCOTT SANNER*, Mechanical and Industrial Engineering, University of Toronto, Canada
KAI LUO, Mechanical and Industrial Engineering, University of Toronto, Canada
GA WU†, Borealis AI, Canada

Critiquing is a method for conversational recommendation that incrementally adapts recommendations in response to user preference feedback. Specifically, a user is iteratively provided with item recommendations and attribute descriptions for those items; the user may then either accept the recommendation or choose to critique an attribute to generate a new recommendation. A recent direction known as latent linear critiquing (LLC) takes a modern embedding-based approach that seeks to optimize the combination of user preference embeddings with embeddings of critiques based on subjective item descriptions (i.e., keyphrases from user reviews); LLC does so by exploiting the linear structure of the embeddings to efficiently optimize their weights in a linear programming (LP) formulation. In this paper, we revisit LLC and note that it's score-based optimization approach inherently encourages extreme weightings in order to maximize predicted score gaps between preferred and non-preferred items. Noting that the overall end task objective in critiquing is to re-rank rather than re-score, in this paper we take a ranking optimization approach that seeks to optimize embedding weights based on observed rank violations from earlier critiquing iterations. We evaluate the proposed framework on two recommendation datasets containing user reviews. Empirical results demonstrate that ranking-based LLC generally outperforms scoring-based LLC and other baselines across a variety of datasets, critiquing styles, and both satisfaction and session-length performance metrics.

Additional Key Words and Phrases: Conversational Recommendation, Critiquing

ACM Reference Format:
Hanze Li, Scott Sanner, Kai Luo, and Ga Wu. 2020. A Ranking Optimization Approach to Latent Linear Critiquing for Conversational Recommender Systems. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 21–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3383313.3412240

1 INTRODUCTION

Critiquing [Chen and Pu 2012] is a method for conversational recommendation that iteratively adapts suggested items according to user feedback regarding desired attributes of those items. Historical critiquing methods focused on modifying recommendations w.r.t. critiques of explicitly known item attributes. The earliest critiquing methods such as unit critiquing [Burke et al. 1996] allowed a user to critique an item recommendation by requesting a change in a specified attribute dimension. Such methods were later extended to compound critiquing [Reilly et al. 2004a, 2005], where a user might further explore items that have logical combinations of unit critiques relative to the current recommendation.

*Faculty Affiliate of the Vector Institute of Artificial Intelligence, Canada.
†The majority of the work was done while the author was at the University of Toronto.

Manuscript submitted to ACM

Further extensions such as incremental critiquing consider the cumulative effect of iterated critiquing interactions [Reilly et al. 2004b] while experience-based methods attempt to collaboratively leverage critiquing interactions from multiple users [McCarthy et al. 2010]. Some other works explored speech- and dialog-based interfaces for critiquing-style frameworks [Grasch et al. 2013; Thompson et al. 2004], however, as previously noted, all of this prior critiquing work assumed explicitly known item attributes.

A more recent direction known as *latent linear critiquing* (LLC) [Luo et al. 2020a] builds on modern embedding-based recommender systems [He et al. 2017; Liang et al. 2018; Sedhain et al. 2016, 2015; Wu et al. 2016] and combines it with an embedding-based approach to critiquing [Antognini et al. 2020; Luo et al. 2020b; Wu et al. 2019]. LLC seeks to optimize the combination of user preference embeddings with embeddings of critiques based on subjective item descriptions (i.e., keyphrases from user reviews); it does so by exploiting the linear structure of the embeddings to efficiently optimize their weights in a linear programming (LP) formulation.

For historical context, LLC has conceptual roots in example-based critiquing [Faltings et al. 2004], which (in the utilitarian view) assumed that critiques would be expressed in terms of feedback based on examples that provided linear constraints on legal penalty functions over item attributes. LLC differs in that critiques are based on keyphrases with latent embeddings (rather than being expressed in terms of explicit attributes or penalties over them); nonetheless, critiques in LLC do induce linear constraints over legal weightings of these embeddings. More recent work on preference elicitation [Sepliarskaia et al. 2018] contributes an optimization problem over linear weightings of embeddings from a latent factor recommendation model; while similar to the embedding-based LLC formulation; the focus of LLC on critiquing with keyphrase co-embeddings leads to a fundamentally different linearly constrained optimization solution.

In this paper, we revisit the LP formulation of the original LLC [Luo et al. 2020a] and observe that its objective maximizes the pairwise difference of rating scores of non-critiqued vs. critiqued items. We argue in this paper that such an objective is problematic because it inherently encourages extreme weightings in order to maximize score gaps between item pairs. Such extreme weights are akin to overfitting in machine learning and we conjecture that a ranking-based LLC approach that instead optimizes latent embedding weights to achieve a desired *rank order* more directly optimizes the end recommendation task of re-ranking items based on critiquing feedback.

We evaluate our proposed ranking-based LLC on two recommendation datasets containing user reviews. Empirical results compared to principled embedding averaging baselines and the existing scoring-based LLC show that our ranking-based LLC generally reduces the number of interactions required to find a satisfactory item and increases the overall percentage of successfully retrieved items. It is also slightly faster than scoring-based LLC and works well with a variety of simulated user critiquing approaches. In summary, this paper provides a novel ranking-based refinement of the LLC framework in a modern conversational recommendation setting involving embedded preferences and language-based keyphrase descriptions that achieves state-of-the-art critiquing performance.

2 PRELIMINARIES

We begin by introducing embedding-based recommendation methods that permit co-embedding of review-based data. These user and review content embeddings will then be leveraged in the latent linear critiquing framework of Section 3.

2.1 Notation

Before proceeding, we define the following notation:

- U is a set of users, I is a set of items, K is a set of keyphrases, and H is a set of hidden (latent) dimensions.

Table 1. Yelp and Beer dataset with examples of extracted keyphrases. We note that the Type column is simply used to organize the table for readability; the Type is not known for arbitrary keyphrases and hence not used in this work.

Dataset	Type	Keyphrases
Yelp	Cuisine	chinese, thai, italian, mexican, vietnamese, japaneses, french
	Food	chicken, beef, fish, pork, seafood, cheese, fried rice
	Drink	tea, coffee, bubble tea, wine, soft drinks, sparkling water
	Price & Service	cheap, pricy, expensive, quick service, busy, friendly
Beer	Head	white, tan, offwhite, brown, mocha
	Malt	roasted, caramel, pale, wheat, rye
	Color	golden, copper, orange, black, yellow, red, ruby
	Taste	citrus, fruit, chocolate, cherry, plum, sweet, honey

- $R \in \mathbb{B}^{|U|\times|I|}$. This is the given binary user preference matrix. Entries $r_{u,i}$ are either 1 (preference observed) or 0 (preference not observed). $\mathbf{r}_u$ represents all feedback from user u, and $\mathbf{r}_{:,i}$ represents all user feedback for item i.
- $M \in \mathbb{R}^{|U|\times|K|}$. 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 users and term frequencies of keyphrases. We use $\mathbf{m}_u$ to represent the uth user's keyphrase frequencies, and $\mathbf{m}_{:,k}$ to represent the kth keyphrase's frequency across all users.
- $M' \in \mathbb{R}^{|I|\times|K|}$. This is the item-keyphrase matrix similar to M except that it aggregates keyphrase frequencies for each item. We use $\mathbf{m}'_i$ to represent ith item's keyphrase frequencies, and $\mathbf{m}'_{:,k}$ to represent kth keyphrase's frequency across all items.
- $Z \in \mathbb{R}^{|U|\times|H|}$. This is the latent user embedding from either **items** or **keyphrases**. We use $\mathbf{z}_u$ to represent uth user's embedding from its *observed preference* and $\mathbf{z}^t_u$ to represent uth user's embedding from its *keyphrase critiquing* at time step t.
- $D \in \mathbb{R}^{|I|\times|H|}$. This is the learned item decoder matrix for PLRec-style matrix factorization covered in Section 3. We use $\mathbf{d}_i$ to represent ith item row in the matrix.
- $E \in \mathbb{R}^{|K|\times|H|}$. This is the learned keyphrase encoder matrix for mapping from user keyphrase space to user latent embedding.
- $i^{-k} \in \{i|M'_{i,k} = 0, \forall i\}$. This item set represents items that do not contain the critiqued keyphrase k.
- $i^{+k} \in \{i|M'_{i,k} > 0, \forall i\}$. This item set represents items that contain the critiqued keyphrase k.
- $c^t_u \in \mathbb{R}^{|1|\times|H|}$. This is the user u specified keyphrase critique c^u_t at each time step $t \in \{1, \cdots, T\}$

2.2 Projected Linear Recommendation

2.2.1 Large-scale Linear Recommender System. Projected Linear Recommendation (PLRec) is a term that we use for the state-of-the-art recommendation method of [Sedhain et al. 2016] that mitigates the scalability problem by projecting preferences from R into a reduced-dimension embedded space prior to linear regression. Formally, the PLRec objective is defined as

$$\underset{D}{\operatorname{argmin}} \sum_u \left\| \mathbf{r}_u - \mathbf{r}_u V D^T \right\|_2^2 + \Omega(W), \tag{1}$$

where decoding parameters D are learned, V is a fixed embedding projection matrix, and Ω is a regularization term. 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 $|H|$

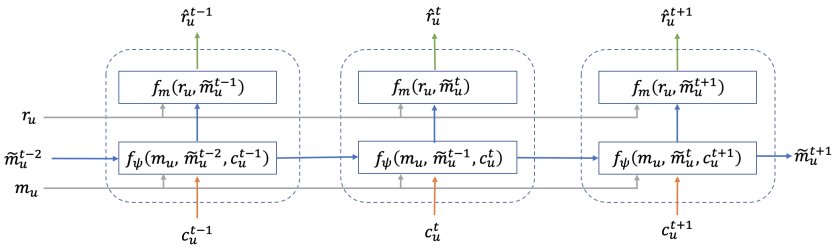


Fig. 1. The flow of conversational critiquing over three time steps. Previously critiqued keyphrases $\tilde{\mathbf{m}}_u^{t-2}$ at time $t-2$ are combined by f_ψ with the newly critiqued keyphrase $\mathbf{c}_u^{t-1}$ at time $t-1$ to yield $\tilde{\mathbf{m}}_u^{t-1}$. The recommendation produced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

of V is far smaller than the observation dimensions $|U|$ and $|I|$. We denote the projected, embedded representation of user u as $\mathbf{z}_u = \mathbf{r}_u V$. Then the prediction score of an interaction between user u and item i is

$$\hat{r}_{u,i} = \langle \mathbf{z}_u, \mathbf{d}_i \rangle , \tag{2}$$

where $\mathbf{d}_i$ is the ith row of D corresponding to item i's latent embedding learned in (1).

2.2.2 Co-embedding of Language-based Feedback. PLRec framework is able to embed language-based feedback in the same space as user preferences. For the recommendation datasets with both preference and review feedback from users, we expect the user preference (or rating) is consistent with the corresponding review text. Based on this assumption, once the latent embedding for a user $\mathbf{z}_u$ is obtained, we can learn to embed keyphrases (extracted from reviews) of a user by training to recover the user's latent preference embedding *from* the preference content implicitly revealed via their reviews.

To make this concrete, for each user u, PLRec encodes their latent preference representation $\mathbf{z}_u$ as in Equation (1). In addition, we have review content for each user represented as a term frequency vector (user keyphrases) $\mathbf{m}_u$. With this, we can now cast the co-embedding task as the following linear regression problem:

$$\underset{E,\mathbf{b}}{\operatorname{argmin}} \sum_u \left\| \mathbf{z}_u - \mathbf{z}'_u \right\|_2^2 + \Omega(E) \tag{3}$$

where

$$\mathbf{z}'_u = \mathbf{m}_u E^T + \mathbf{b} \tag{4}$$

is the initial user keyphrase embedding and $E \in \mathbb{R}^{|K| \times |H|}$ is the learned keyphrase encoder that projects users' review text 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 the next section.

3 LATENT LINEAR CONVERSATIONAL CRITIQUING

We now proceed to recap the scoring-based latent linear critiquing (LLC) framework [Luo et al. 2020a] that we extend in this work. In the LLC framework, 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.

Each critiquing step can be viewed as a series of functional transformations that produce a modified prediction $\hat{\mathbf{r}}_u$ of item preferences for user u given critiqued item keyphrases $\tilde{\mathbf{m}}_u^t$ at time step t as follows:

$$\hat{\mathbf{r}}_u^t = f_\phi(\mathbf{r}_u, \tilde{\mathbf{m}}_u^t), \quad given \quad \tilde{\mathbf{m}}_u^t = f_\psi(\mathbf{m}_u, \tilde{\mathbf{m}}_u^{t-1}, \mathbf{c}_u^t), \tag{5}$$

where the critique-modified recommendation function f_ϕ takes user preferences $\mathbf{r}_u$ and critiqued keyphrases $\tilde{\mathbf{m}}_u^t$ at time step t as input and produces a recommendation $\hat{\mathbf{r}}_u^t$ as output. The function f_ψ updates the critiqued keyphrases $\mathbf{m}_u^{t-1}$ by applying a user critiquing action $\mathbf{c}_u^t$ to user keyphrases $\mathbf{m}_u$.

In practice, an actual critiquing method will need to provide concrete definitions of these abstract functions f_ϕ and f_ψ. One such framework is LLC, covered next.

3.1 Latent Linear Critiquing

Latent Linear Critiquing (LLC) [Luo et al. 2020a] is a critiquing-based recommendation framework that specifies how to make improved recommendations after a user u has provided critiques $\mathbf{c}_u^1 \cdots \mathbf{c}_u^t$ over T iterations (as demonstrated in Figure 1). Here, critiques $\mathbf{c}_u^t$ are encoded as one-hot keyphrase indicators that represent a user u's *dislike* of a keyphrase description at time step t.

To transform the user's critiques $\mathbf{c}_u$ into an embeddable term frequency representation that can be co-embedded with user preference embeddings, LLC defines the cumulative critiquing function f_ψ as follows:

$$\tilde{\mathbf{m}}_u^t = f_\psi(\mathbf{m}_u, \tilde{\mathbf{m}}_u^{t-1}, \mathbf{c}_u^t) = \tilde{\mathbf{m}}_u^{t-1} - \max(\mathbf{m}_u, \mu(\mathbf{m}_u)) \odot \mathbf{c}_u^t \tag{6}$$

where $\odot$ represents element-wise multiplication of two vectors and the initial $\tilde{\mathbf{m}}_u^0$ is a zero vector of length $|K|$.

With the critiqued keyphrases $\mathbf{m}_u^t$ and the mapping between keyphrase and user latent representation learned in Equation (3), LLC provides a latent representation of *all* critiques in matrix form

$$Z_u^t = diag(\tilde{\mathbf{m}}_u^t)E^T + B \tag{7}$$

where each row $\mathbf{z}_u^t$ of the matrix Z_u^t represents the latent representation of the tth critiqued keyphrase, and each row of B is the identical bias term $\mathbf{b}$.

As there exist multiple user representations, the LLC framework proposed to define a blending function that aggregates all embeddings through a linear combination such that

$$\phi_{\boldsymbol{\theta}}(\mathbf{z}_u, Z_u^t) = \theta_0 \mathbf{z}_u + \theta_1 \mathbf{z}_u^1 + \cdots + \theta_T \mathbf{z}_u^T. \tag{8}$$

The previous work on LLC [Luo et al. 2020a] defined two principled heuristic averaging options to be used as baseline methods to assign coefficients $\boldsymbol{\theta}$:

- **Uniform Average Critiquing** averages the user preferences and critiqued keyphrase embeddings uniformly:

$$\phi_{\boldsymbol{\theta}}(\mathbf{z}_u, Z_u^t) = \frac{1}{T+1}(\mathbf{z}_u + \mathbf{z}_u^1 + \cdots + \mathbf{z}_u^T). \tag{9}$$

- **Balanced Average Critiquing** averages critique embeddings Z_u together and then averages them again with the user preference embedding, thereby balancing the critique and user embeddings:

$$\phi_{\boldsymbol{\theta}}(\mathbf{z}_u, Z_u^t) = \frac{1}{2}\left(\mathbf{z}_u + \frac{1}{T}\left(\mathbf{z}_u^1 + \cdots \mathbf{z}_u^T\right)\right). \tag{10}$$

3.2 Scoring-based Latent Linear Critiquing

A key drawback of the previously defined heuristic methods for uniform and balanced average critiquing is that there is no inherent guarantee that such simple averages of preference and critique embeddings will necessarily lead to a drop in rank for items likely to have the critiqued keyphrase description. In order to derive a more optimal weighting, scoring-based LLC [Luo et al. 2020a] made a few critical observations that paved the way for an optimization approach. First, it observed that the form of Equation (8) is linear in the parameters θ, which facilitates efficient linear optimization approaches. Second, while the entire point of the latent critiquing framework is that latent (not explicitly known) attributes of items can be critiqued, it pointed out that we know that some items i^+ are explicitly described by the critiqued keyphrase while we may infer that other items i^- are unlikely to be described by the keyphrase.

Hence, at each time step t, the scoring-based LLC work set out to optimize the θ's such that the scores (and hence ranks) of critiqued items i^+ will fall while the scores of non-critiqued items i^- are likely to rise. Formally, the weights θ can be optimized by the following linear programming (LP) optimization problem:

$$\begin{aligned} &\max_{\theta_0,\ldots,\theta_T} \sum_{i^+} \sum_{i^-} \left(\hat{r}^t_{u,i^-} - \hat{r}^t_{u,i^+} \right) \\ &\text{subject to: } \theta_0 = 1 \\ &\qquad\qquad \theta_t \in [-1,+1] \; \forall t \in \{1,\ldots,T\} \end{aligned} \tag{11}$$

In the above LP formulation, user and critique embedding weights $\theta_0, \ldots, \theta_T$ are optimized to maximize the pairwise difference of ratings of non-critiqued items i^- with critiqued items i^+. To limit the computational complexity, scoring-based LLC considers the top-K ($K = 100$ by default) rated items meeting the criteria for i^+ and i^-.

The constraints of (11) are rather simple, but are reported to perform the best among a few options considered in [Luo et al. 2020a]. In short, they enforce that the user preference embedding weight (θ_0) is always fixed at 1, while all of the critique embedding weights θ_t for $t \in \{1,\ldots,T\}$ are allowed to vary in the range $[-1,1]$ relative to the fixed θ_0.

We remark that $\hat{r}^t_{u,i^\pm}$ in the objective are linear functions of $\theta_0, \ldots, \theta_T$ and hence the overall framework in (11) is clearly a linear program (LP). Overall, while this scoring-based LLC framework is highly scalable due to the efficiency of modern LP solvers and effective in practice as we will show empirically, it suffers from one major design flaw. That is, the nature of the score gap maximization in the objective and the simple bound constraints naturally lead to extreme weights — by definition, the optimal weights should generally be at the ± 1 extremes that define the vertices of the convex constraint polytope. We conjecture in this paper that such extreme weights may overall be harmful for strong performance and seek to mitigate this concern with our ranking-based LLC framework introduced in the next section.

4 RANKING-BASED LATENT LINEAR CRITIQUING

Since our overall goal in the LLC framework is to re-rank the recommended items based on provided critique embeddings, we now seek to re-envision LLC from a ranking perspective rather than the previous scoring-based perspective.

4.1 Incremental Optimization Formulation

Quite simply, at any iteration of critiquing τ, we can assume that $\boldsymbol{\theta}^{\tau-1}$ represents the weighting used to generate the current recommendations and $\boldsymbol{\theta}^{\tau}$ is a new weighting that we wish to optimize subject to the latest critique $\boldsymbol{c}^{\tau}_u$. Furthermore, given the user's critique at iteration τ, like scoring-based LLC we can obtain a set of non-critiqued items i^- and critiqued items i^+ within the top-K recommendation ranks from iteration $\tau - 1$. Here, we want a new weighting $\boldsymbol{\theta}^{\tau}$ that improves relative to $\boldsymbol{\theta}^{\tau-1}$, where the *rank* of items i^- should increase while the rank of i^- should decrease.

We can express this overall ranking-based objective via a simple pairwise score-comparison objective motivated by RankSVM [Joachims 2002]. Like RankSVM, we introduce slack variables ξ representing rank violations of pairwise preferences. These ξ should be minimized in the objective along with a bias towards the agnostic uniform averaging critiquing solution in case all rank preferences can be satisfied:

$$\begin{aligned}
\min_{\boldsymbol{\theta}^\tau} \quad & V(\boldsymbol{\theta}^\tau, \xi) = \left\| \boldsymbol{\theta}^\tau - \frac{1}{T+1} \right\|_1 + \lambda \sum_{i=1}^{|I|} \xi_i \\
\text{subject to:} \quad & \forall \tau \in \{1 \cdots T\}, i^+ : \langle \phi_{\boldsymbol{\theta}^{\tau-1}}(\mathbf{z}_u, Z_u^{\tau-1}), \mathbf{d}_{i^+} \rangle > \langle \phi_{\boldsymbol{\theta}^\tau}(\mathbf{z}_u, Z_u^\tau), \mathbf{d}_{i^+} \rangle + 1 - \xi_{i^+} \\
& \forall \tau \in \{1 \cdots T\}, i^- : \langle \phi_{\boldsymbol{\theta}^\tau}(\mathbf{z}_u, Z_u^\tau), \mathbf{d}_{i^-} \rangle > \langle \phi_{\boldsymbol{\theta}^{\tau-1}}(\mathbf{z}_u, Z_u^{\tau-1}), \mathbf{d}_{i^-} \rangle + 1 - \xi_{i^-} \\
& \forall i : \xi_i \geq 0
\end{aligned} \tag{12}$$

We now pause to explain some additional details of this ranking-based linear programming (LP) formulation:

- At time step 0, $\theta_0 = 1$, $Z_u^0 = \emptyset$ (abusing notation), and hence $\phi_{\boldsymbol{\theta}}(\mathbf{z}_u, Z_u^0) = \mathbf{z}_u$. At time step τ, $\boldsymbol{\theta}^{\tau-1}$ is the *constant* weight vector recorded from the previous critiqued time step that is no longer in the scope of optimization.
- i^+ represent affected items (known to have the critiqued keyphrase) and i^- represent unaffected items. The LHS in the affected item i^+'s equation represents the previous time step's item score using $\boldsymbol{\theta}^{\tau-1}$ while the RHS represents the current time step's (post-critiquing) item score using $\boldsymbol{\theta}^\tau$ after critique $\boldsymbol{c}_u^\tau$ — we want this new RHS item score to drop. The case is reversed for i^- where we want it's item score to increase.
- ξ_i is a non-negative slack variable used in the RankSVM formulation for the score constraint on item i. If a score constraint is satisfied then ξ_i can be minimized to 0 and ignored in the objective. If the score constraint is violated then ξ_i represents the amount of violation (penalized in the objective). Similar to the RankSVM which only placed constraints on scores, this formulation is not guaranteed to change the score (or rank) of all constrained items — the ξ_i can absorb violations when all pairwise score constraints cannot be mutually satisfied.
- As noted before, we introduce the $\|\cdot\|_1$ norm preferring the average weighting of embeddings in the case that the pairwise constraints can all be achieved and there is no other optimization guidance on which weights to choose. It also serves in a mild regularization role. The piecewise linear absolute value in the $\|\cdot\|_1$ portion of the objective can be converted to a purely linear objective using standard mathematical programming transformations.
- λ is the regularization coefficient for trading-off between the preference for the average weighted solution and a larger penalty on rank violations to more strongly enforce the pairwise comparison constraints. We treat this as a hyperparameter to be tuned in the experiments.

If we now take a closer look at the constraints, we note that the $\phi_{\boldsymbol{\theta}}(\mathbf{z}_u, Z_u)$ are a weighted sum of constant embeddings for the user, item, and keyphrase critique embeddings that are linear in the free parameters $\boldsymbol{\theta}^\tau$. To see this more clearly, we explicitly expand the constraints in (12):

$$\begin{aligned}
& \forall \tau \in \{1 \cdots T\}, i^+ : \langle \theta_0^{\tau-1}\mathbf{z}_u + \theta_1^{\tau-1}\mathbf{z}_u^1 \cdots \theta_{t-1}^{\tau-1}\mathbf{z}_u^{\tau-1}, \mathbf{d}_{i^+} \rangle > \langle \theta_0^\tau \mathbf{z}_u + \theta_1^\tau \mathbf{z}_u^1 \cdots \theta_t^\tau \mathbf{z}_u^\tau, \mathbf{d}_{i^+} \rangle \rangle + 1 - \xi_{i^+} \\
& \forall \tau \in \{1 \cdots T\}, i^- : \langle \theta_0^\tau \mathbf{z}_u + \theta_1^\tau \mathbf{z}_u^1 \cdots \theta_t^\tau \mathbf{z}_u^\tau, \mathbf{d}_{i^-} \rangle > \langle \theta_0^{\tau-1}\mathbf{z}_u + \theta_1^{\tau-1}\mathbf{z}_u^1 \cdots \theta_{t-1}^{\tau-1}\mathbf{z}_u^{\tau-1}, \mathbf{d}_{i^-} \rangle + 1 - \xi_{i^-}
\end{aligned} \tag{13}$$

Here, the $0, \ldots, t$ subscripts to θ's are indices for each scalar element inside $\boldsymbol{\theta}$; furthermore, the $\mathbf{z}_u$ is the initial user preference embedding while $\mathbf{z}_u^1$ to $\mathbf{z}_u^{\tau-1}$ represent the embeddings for each of the $\tau - 1$ keyphrase critiques stacked in matrix Z_u. As for the scoring-based LLC, we can see that our constraints are still linear and as noted earlier, despite the absolute value in the $\|\cdot\|_1$ term of the objective, it can also be converted to a linear form. Hence, we obtain an alternative, but still efficiently optimizable and highly scalable linear programming (LP) formulation.

While this resulting LP optimization framework may seem similar to the LP of the scoring-based LLC, we note that $\boldsymbol{\theta}^\tau$ need only be optimized enough to shift its relative score — because it is a pairwise constraint to be satisfied, there is no additional reward for the *degree* to which it is satisfied. As noted in the previous motivation for this method, we conjecture that this pairwise constraint approach of ranking-based LLC places less pressure on the optimization to use extreme weightings (that occurs in scoring-based LLC and which may hurt empirical performance) while still nudging all affected item scores in the intended direction to respect the user critiques.

4.2 Non-incremental Optimization Variant

One might consider that the ranking optimization in Equation (12) has a moving target that incrementally determines the best $\boldsymbol{\theta}^\tau$ w.r.t. the previous best $\boldsymbol{\theta}^{\tau-1}$. This is contrary to the much simpler non-incremental formulation of the original RankSVM and in that spirit, we now consider a non-incremental variation of Equation (12) that always attempts to find the best $\boldsymbol{\theta}^\tau$ for all accumulated rank preference constraints of critiques up to iteration τ relative to the original critique-free personalized preference embedding for the user. We conjecture that this might lead to a simpler and more stable result though we defer to the empirical evaluation of Section 5.4 for the final verdict on this alternative.[1]

$$\begin{aligned} \min_{\boldsymbol{\theta}^\tau} \quad & V(\boldsymbol{\theta}^\tau, \xi) = \left\| \boldsymbol{\theta}^\tau - \frac{1}{T+1} \right\|_1 + \lambda * \sum_{i=1}^{|I|} \xi_i \\ \text{subject to:} \quad & \forall \tau \in \{1 \cdots T\}, i^+ : \langle \mathbf{z}_u, \mathbf{d}_{i^+} \rangle > \langle \phi_{\boldsymbol{\theta}}(\mathbf{z}_u, Z_u^\tau), \mathbf{d}_{i^+} \rangle + 1 - \xi_{i^+} \\ & \forall \tau \in \{1 \cdots T\}, i^- : \langle \phi_{\boldsymbol{\theta}}(\mathbf{z}_u, Z_u^\tau), \mathbf{d}_{i^-} \rangle > \langle \mathbf{z}_u, \mathbf{d}_{i^-} \rangle + 1 - \xi_{i^-} \\ & \forall i : \ \xi_i \geq 0 \end{aligned} \tag{14}$$

In comparing this to the original formulation of Equation (12), we note that these pairwise rank constraints in Equation (14) only compare the current step's $\boldsymbol{\theta}^\tau$-based item score to the initial item score based solely on the user preference embedding $\langle \mathbf{z}_u, \mathbf{d}_{i^+} \rangle$.

5 EXPERIMENTS

In the experiment section, we proceed to evaluate the proposed ranking-based LLC in order to answer the following research questions:

(1) Does our proposed ranking-based LLC algorithm perform better than other baselines and scoring-based LLC for different methods of critiquing keyphrase selection, datasets, and metrics?
(2) What is the empirical time complexity for our proposed algorithm compared to scoring-based LLC? Does our proposed algorithm consume more or less computation time?
(3) Does the Incremental optimization approach previously proposed for ranking-based LLC outperform the Non-incremental version?

Code to reproduce all of the following experiments is on github[2].

5.1 Experiment Settings

5.1.1 Dataset. We evaluate the proposed ranking-based LLC framework on two publicly available datasets: BeerAdvocate [McAuley et al. 2012] and our own private crawl of the Yelp website. Each of the datasets contains more than

[1] Spoiler alert: it does not actually perform better.
[2] https://github.com/litosly/RankingOptimizationApprochtoLLC

Table 2. Summary of datasets.

Dataset	# Users	# Items	Rating Sparsity	Keyphrase Coverage	Keyphrase Average Counts (per User)
Yelp	2,343	7,456	0.2115%	99.11%	9.9248
Beer (BeerAdvocate)	6,370	3,668	1.1268%	99.29%	55.1088

Algorithm 1 User Simulation Evaluation

procedure EVAL($\bar{R}$ for test)
  for each user u **do**
    for each target item i, where $\bar{r}_{u,i} = 1$ **do**
      for time step $t \in range(1, MAX)$ **do**
        user act critique $\mathbf{c}^t$
        determine $\theta_0 ... \theta_t$ using UAC, BAC, LLC-Rank, or LLC-Score
        $\hat{r}^t_{u,i} \leftarrow \langle \theta_0 \mathbf{z}_i + \theta_1 (\mathbf{z}_i)^1 + \cdots \theta_T (\mathbf{z}_i)^T, \mathbf{d}_i \rangle$
        if i in Top-N recommendation list **then**
          break session with success
      length $\leftarrow min(t, MAX)$
  return average success rate & length

100,000 reviews and product rating records. For the purpose of Top-N ranking evaluation, we binarize the rating column of both datasets with a rating threshold ϑ. In Yelp, the threshold is $\vartheta > 3$ out of 5. Due to the fact that users tend to rate positively in BeerAdvocate, we define the rating threshold $\vartheta > 4$ out of 5. Table 2 shows overall dataset statistics for our experiments. Unlike [Luo et al. 2020a], we do not evaluate on Amazon CDs & Vinyl, where we obtained initially poor results, but also noticed unusually high keyphrase coverage for items suggesting the need for improved data cleaning.

5.1.2 Automated Keyphrase Extraction. Recommendation datasets typically do not come with pre-selected keyphrases to describe either user or items. Thus, we choose to obtain the keyphrases directly from user reviews, where the keyphrases are used for explanation and critiquing. While there is a reasonable concern regarding the general quality of reviews for supporting recommendation performance (cf. [Sachdeva and McAuley 2020]), we assume a positive correlation exists between historical item preferences and collective review content used to describe those items. Hence, we used the following generic processing steps to extract candidate keyphrases from the reviews:

(1) Extract separate unigram and bigram lists of high frequency noun and adjective phrases from dataset reviews.
(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. From this information, construct both the user-keyphrase matrix M and item-keyphrase matrix M'.

5.1.3 User Simulation for Critiquing Performance 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 followed with given keyphrase selection method, 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 5 simulated sessions and estimate the average success rate as well as the average session length for model comparison. The Top-N ranking threshold in this experiment is selected from $\{1, 5, 10, 20\}$, where success becomes easier with increasing N. The maximum allowed critiquing iterations in Algorithm 1 is set to 10.

In order to simulate a variety of user critiquing styles, we experimented with three different keyphrase selection methodologies:

- **Random:** We assume the user **randomly chooses** a keyphrase to critique that is inconsistent with the target item's known keyphrase list.
- **Diff:** We assume the user may prefer to critique a keyphrase that **deviates the most** from the known target item description. During simulation, this is done by comparing the top recommended items' keyphrase frequency to the target item's keyphrase frequency and then critiquing the keyphrase with the largest frequency differential.
- **Pop:** We assume the user will select a keyphrase to critique based on general keyphrase popularity. More specifically, the user will critique **the most popular keyphrase** used across all reviews that is inconsistent with the target item's known keyphrase list.

All keyphrase critique selection methods naturally prevent critiquing the same keyphrase multiple times in the same user simulation session.

Two example user critiquing simulations with our proposed ranking-based LLC approach are shown in Table 6 and discussed in Section 5.5.

5.1.4 Candidate Latent Critiquing Algorithms. We experiment with the following previously described latent critiquing methods that choose the weight of the user and critiqued keyphrase embeddings in our LLC framework:

- **UAC:** Uniform Average Critiquing from Equation (9).
- **BAC:** Balanced Average Critiquing from Equation (10).
- **LLC-Score:** The current scoring-based LLC algorithm [Luo et al. 2020a] that maximizes the rating score difference between critiqued and non-critiqued items as optimized in Equation (11).
- **LLC-Rank:** Our proposed LLC algorithm using the ranking-based optimization approach from Equation (12).

5.2 Critiquing Performance Evaluation

In this section, we answer the question of how our proposed ranking optimization approach performs compared to baseline methodologies described in Section 5.1.4. In particular, we conduct user simulation experiments described above in Section 5.1.3 and evaluate the proposed method along with multiple baselines using two metrics: *Average Success Rate* (the average number of sessions for a user that terminate with success) and *Session Length* (the average length of a user session with a maximum session length of 10 critiquing iterations).

We remark that different from the experimental settings in [Luo et al. 2020a], we do not constrain refined recommendation results at an iteration to only include items that did not appear in the Top-N items in previous iterations. Overall, this results in a relatively lower success rate and larger average session length than reported in [Luo et al. 2020a]. In our experimental framework, we believe that the recommendation system would not likely know the desired rank threshold N of the user and thus opted for this change in order to make for a more realistic simulation.

Table 3. Average Success Rate and Average Session Length for recommendation targeting at Top-N (1,5,10,20) with 95% confidence intervals. ***Random*** Keyphrase Selection Method. Higher is better for Average Success Rate; Lower is better for Average Session Length.

		Yelp				Beer			
	@Top-N	1	5	10	20	1	5	10	20
Average Success Rate	UAC	0.0046±0.003	0.0369±0.012	0.0561±0.016	0.0830±0.018	0.0171±0.023	0.0274±0.025	0.0686±0.037	0.1060±0.041
	BAC	0.0036±0.003	0.0309±0.011	0.0475±0.015	0.0719±0.017	0.0114±0.013	0.0261±0.025	0.0556±0.03	0.0936±0.039
	LLC-Score	0.0141±0.007	0.0479±0.014	0.0689±0.017	0.0943±0.02	0.0114±0.013	**0.0416±0.023**	0.0656±0.033	0.1174±0.043
	LLC-Rank	**0.0161±0.008**	**0.069±0.017**	**0.0893±0.02**	**0.1224±0.023**	**0.0183±0.023**	0.0375±0.027	**0.0704±0.037**	**0.1204±0.044**
Average Session Length	UAC	9.964±0.027	9.701±0.103	9.545±0.135	9.308±0.154	6.44±0.492	6.396±0.495	6.301±0.502	6.208±0.511
	BAC	9.968±0.025	9.723±0.096	9.580±0.131	9.357±0.150	6.446±0.490	6.396±0.495	6.315±0.498	6.217±0.511
	LLC-Score	9.915±0.049	9.643±0.113	9.484±0.137	9.253±0.158	6.446±0.490	**6.366±0.491**	**6.303±0.498**	**6.149±0.517**
	LLC-Rank	**9.905±0.052**	**9.542±0.12**	**9.363±0.146**	**9.123±0.161**	**6.431±0.492**	6.400±0.497	6.304±0.501	6.151±0.520

Table 4. Average Success Rate and Average Session Length for recommendation targeting at Top-N (1,5,10,20) with 95% confidence intervals. ***Diff*** Keyphrase Selection Method. Higher is better for Average Success Rate; Lower is better for Average Session Length.

		Yelp				Beer			
	@Top-N	1	5	10	20	1	5	10	20
Average Success Rate	UAC	0.0066±0.004	0.0353±0.012	0.0533±0.015	0.0860±0.018	0.0286±0.032	0.0528±0.035	0.0792±0.039	0.1093±0.042
	BAC	0.0040±0.003	0.0322±0.011	0.0460±0.014	0.0755±0.017	0.0114±0.013	0.0268±0.025	0.0543±0.031	0.0927±0.040
	LLC-Score	0.0054±0.005	0.0322±0.011	0.0496±0.015	0.0761±0.017	0.0114±0.013	0.0396±0.023	0.0634±0.033	0.1073±0.041
	LLC-Rank	**0.0196±0.009**	**0.0585±0.016**	**0.0849±0.019**	**0.134±0.024**	**0.0312±0.021**	**0.0579±0.027**	**0.0892±0.032**	**0.1773±0.05**
Average Sesssion Length	UAC	9.955±0.029	9.711±0.098	9.573±0.128	9.277±0.157	**9.806±0.216**	9.622±0.263	**9.349±0.331**	9.075±0.359
	BAC	9.965±0.026	9.715±0.096	9.593±0.126	9.326±0.153	9.897±0.116	9.758±0.224	9.513±0.275	9.184±0.343
	LLC-Score	9.952±0.044	9.715±0.099	9.563±0.129	9.329±0.152	9.897±0.116	9.648±0.210	9.431±0.293	9.041±0.365
	LLC-Rank	**9.896±0.052**	**9.627±0.105**	**9.438±0.128**	**9.098±0.158**	9.826±0.136	**9.565±0.214**	9.396±0.241	**8.774±0.384**

5.2.1 Optimization Methods Comparison. We start by evaluating whether the proposed rank-based optimization approach outperforms the baseline methods (i.e. UAC and BAC) and scoring-based LLC under various experimental settings and metrics. Tables 3, 4, and 5 show the percentage of target items that successfully reach a rank of N $\in \{1, 5, 10, 20\}$ before the session terminates along with the average length of these sessions (both results averaged over users). We grouped the experimental results by keyphrase selection methods.

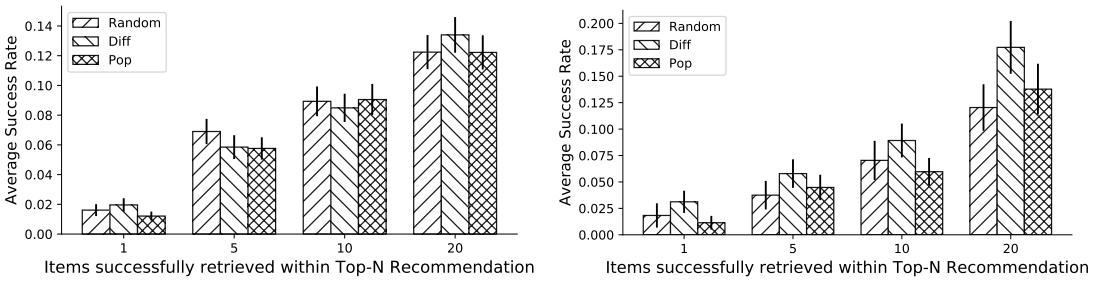
Our proposed incremental ranking optimization approach consistently outperforms the baseline methods (i.e. UAC, BAC , and LP-rating) in both datasets for most metrics. Perhaps the most notable result is that in some cases (cf. Table 4) LLC-Rank *triples* performance over the nearest competing method — including LLC-Score – on the average success rate. The average critiquing session length results reflect the success rate results in general — a higher success rate usually leads to earlier termination of sessions. However, since the algorithm could terminate at the iteration threshold, i.e. 10 iterations, the algorithm with a lower average success rate does not necessarily terminate later than an algorithm with a higher success rate. We also remark that significant changes in the success rate are not reflected as clearly in the average session length since the success rates are expected to be small for low N and thus many sessions naturally extend to their full length in this case.

We also observe that for the Beer Advocate dataset, the total number of available keyphrases is limited compared to Yelp (we selected 75 keyphrases for Beer Advocate vs. 235 keyphrases for Yelp). Hence, the average session length for Beer Advocate is also shorter given the same maximum iteration threshold allowed, thus leading to a smaller gap in performance on the average session length metric for Beer compared to results for Yelp.

5.2.2 Impact Analysis of Keyphrase Selection Methods. We would also like to further understand the performance of the proposed ranking-based LLC approach vs. the different types of keyphrase critique selection methods in the user

Table 5. Average Success Rate and Average Session Length for recommendation targeting at Top-N (1,5,10,20) with 95% confidence intervals. ***Pop*** Keyphrase Selection Method. Higher is better for Average Success Rate; Lower is better for Average Session Length.

		Yelp				Beer			
	@Top-N	1	5	10	20	1	5	10	20
Average Success Rate	UAC	0.0071±0.005	0.0383±0.012	0.0565±0.015	0.0894±0.018	**0.0171±0.023**	0.0281±0.025	**0.0669±0.037**	0.1036±0.041
	BAC	0.0036±0.003	0.0309±0.011	0.0483±0.015	0.0791±0.017	0.0114±0.013	0.0268±0.025	0.0556±0.030	0.0927±0.040
	LLC-Score	0.0071±0.005	0.0433±0.014	0.0652±0.016	0.0941±0.020	0.0114±0.013	0.0416±0.023	0.0630±0.032	0.1120±0.042
	LLC-Rank	**0.0121±0.006**	**0.0576±0.015**	**0.0905±0.021**	**0.1222±0.023**	0.0114±0.013	**0.0448±0.024**	0.0597±0.026	**0.1378±0.048**
Average Session Length	UAC	9.95±0.0350	9.707±0.098	9.550±0.130	9.249±0.156	**6.44±0.492**	6.395±0.495	**6.307±0.501**	6.215±0.511
	BAC	9.969±0.025	9.730±0.094	9.580±0.128	9.303±0.156	6.446±0.490	6.395±0.495	6.317±0.498	6.221±0.511
	LLC-Score	9.948±0.044	9.662±0.113	9.518±0.130	9.230±0.157	6.446±0.490	**6.359±0.490**	6.308±0.498	**6.160±0.520**
	LLC-Rank	**9.948±0.028**	**9.592±0.116**	**9.381±0.151**	**9.087±0.174**	6.457±0.487	6.374±0.489	6.337±0.491	6.211±0.498



(a) Yelp Dataset (b) Beer Dataset

Fig. 2. LLC-Rank performance under different keyphrase selection methodologies with 95% confidence intervals. Higher is better.

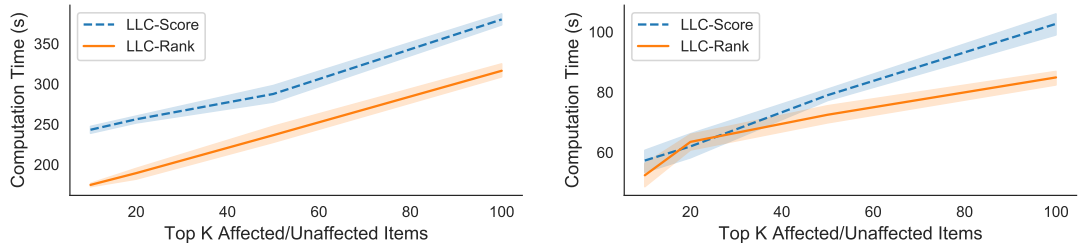
simulation. In Figure 2, we compare our proposed method's performance under different user simulation settings where users *randomly* ("**Random**") select available keyphrases, select keyphrases based on *keyphrase popularity* ("**Pop**"), or select the keyphrase with the *largest keyphrase frequency difference* ("**Diff**") from their ideal target items.

It can be seen that in general, the *Diff* keyphrase selection method provides a higher success rate for our proposed methodology. This matches our expectation for this setting since *Diff* assumes users have a good knowledge of how their target item contrasts with the current recommended item and are able to clearly articulate a discriminative keyphrase critique. Such a keyphrase selection clearly distinguishes the target item from the current recommended items leading to the observed strong performance. It can also be noticed that the *Random* keyphrase selection method has similar performance to selecting keyphrases based on popularity (*Pop*). This indicates that our algorithm can perform relatively well on optimizing weights even with non-ideal critique selection strategies.

5.3 Computation Time Analysis

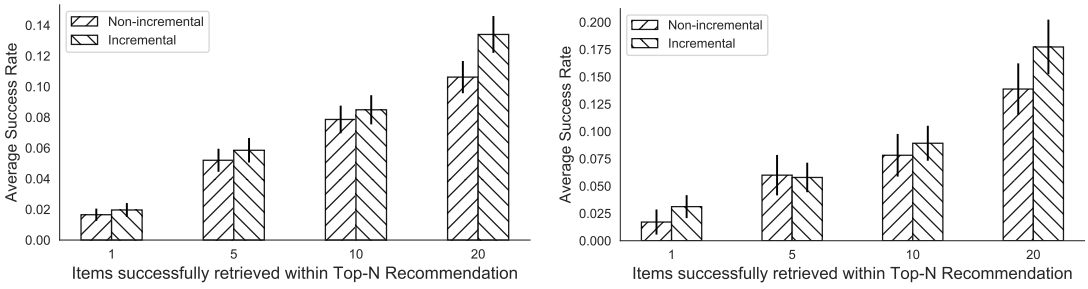
In this section, we determined how the newly proposed LLC-Rank compares to the existing LLC-Score in terms of computation time. Because both LLC-Rank and LLC-Score rely on different optimization frameworks, an analytical worst-case analysis would not necessarily be indicative of average case performance. Hence, in order to assess computation time, we provide an empirical comparison.

Figure 3 shows the computation time in seconds for 10 sessions of ranking-based LLC and scoring-based LLC for the experiments performed in Section 5.2.1 using a *MacBook Pro. 16-inch with 2.6GHz 6-Core Processor and AMD Radeon Pro 5300M*. Both algorithms must select their affected and unaffected (i.e., critiqued and non-critiqued) items for optimization



(a) Yelp Dataset (b) Beer Dataset

Fig. 3. Average time consumed for completing 10 runs for 100-user simulation with 95% confidence intervals. Lower is better.



(a) Yelp Dataset (b) Beer Dataset

Fig. 4. Incremental vs. Non-incremental optimization performance under *diff* keyphrase selection methodologies with 95% confidence intervals. Higher is better.

from a list of the current top K items (K is a different constrant from the user's ranking threshold N). Since a larger K inherently leads to more constraints, we plot performance for both algorithms vs. this K parameter. Overall, we observe that LLC-Rank does appear to be slightly faster overall.

5.4 Incremental vs. Non-incremental Optimization

When we introduced our ranking-based LLC optimization approach in Section 4, we provided an Incremental method (Section 4.1) that sought to correct rankings of critiqued items relative to the weighting that initially yielded that ranking. One might consider though that the ranking objective has a moving target and that a simple accumulation of pairwise rankings compared to all weight on the initial user preference embedding might lead to a simpler and more stable result. This latter method was termed Non-incremental (Section 4.2).

In order to show the benefit of our proposed Incremental ranking optimization approach, we compared it to the Non-Incremental approach in Figure 4. These results show that under the *Diff* keyphrase selection methodology, the Incremental optimization approach consistently outperforms the alternative Non-incremental approach on both the Yelp and Beer Advocate datasets. This matches our expectation that our Incremental approach is a better optimization approach for sequential recommendation and that it more carefully optimizes the embedding weightings by correcting specifically for errors made by previous weight choices.

Table 6. User Case Study with target rank N=1 on the Yelp and Beer Advocate datasets.

Dataset	Time Step t	Recommended Item	Top Keyphrases Describing the Item	Critiqued Keyphrase	Successfully Retrieved?
Yelp	0	Playa Cabana	taco, mexican, chip	taco	-
	1	Banh Mi Boys	pork, fry, sandwich	pork	-
	2	Uncle Tetsu's Japanese Cheesecake	cake, cheese, japanese	cake	-
	3	Kekou Gelato House	gelato, tea, milk	-	✓
Beer	0	Bell's Oberon Ale	wheat, citrus, orange	orange	-
	1	Sanctification	sour, white, lemon	sour	-
	2	Wisconsin Belgian Red	cherry,sweet,red	cherry	-
	3	Snapperhead IPA	gold, sweet, yeast	-	✓

5.5 Case Study

To qualitatively evaluate the performance of the critiquing in a real scenario, we simulated multiple use cases of the proposed methods on the both review datasets. Table 6 shows two representative cases we encountered during our investigation with the *Diff* keyphrase selection methodology.

For the Yelp dataset, based on simulated interactions for a user, we see that the algorithm initially recommends *Playa Cabana* to the user, which is a conventional Mexican restaurant featuring tacos and (tortilla) chips. The user, however, has the target of a Chinese style dessert restaurant *Kekou Gelato House* which offers gelato and milk-tea. Therefore, the user first critiques the keyphrase describing his target item that differs the most from current recommended restaurant, which is "taco". Similarly for iteration 1 and 2, the user critiques "pork" and "cake" before the algorithm converges on the user's target restaurant, clearly aided by the user's historical preferences for desserts as evidenced by the two dessert recommendations at time steps $t = 2$ and $t = 3$. Hence this serves as an excellent example of combining a user's historical preference embedding with their critique embeddings.

The Beer Advocate example is slightly more straightforward. The user's target item of *Snapperhead IPA* is a sweet, non-fruity beer. The *Diff* critiquing method does an excellent job of selecting "orange", "sour", and "cherry" that clearly contrast with the target item and lead to the suggestion of the intended user's target. While such early convergence to the target is undoubtedly partially luck, this simulation demonstrates the adequacy of the user simulation and *Diff* keyphrase critique selection methodology along with iterative recommendations that clearly capture the intent of the keyphrase critiques and appropriately combine them with a user's historical latent preferences.

6 CONCLUSION

In this paper, we extended an existing latent linear critiquing (LLC) framework for multi-step conversational recommendation with a ranking-based optimization approach. This framework permits keyphrase critiques and optimizes their co-embedding in the same latent space as user preferences, thus allowing these general language-based critiques to modulate future item recommendations. Unlike the previous scoring-based LLC approach [Luo et al. 2020a], our new ranking-based approach focused directly on the end task of re-ranking recommended items based on user critique feedback. Our empirical results demonstrated that this direct ranking-based approach generally increases the overall percentage of successfully retrieved items (in some cases *tripling* performance over the nearest competitor), slightly reduces the average session length, and runs faster than scoring-based LLC and other baselines. These results validate our intuition that the re-ranking approach more directly reflects the objective of the end critiquing task, hence leading to stronger overall performance than the previous scoring-based LLC approach.

REFERENCES

Diego Antognini, Claudiu Musat, and Boi Faltings. 2020. T-RECS: a Transformer-based Recommender Generating Textual Explanations and Integrating Unsupervised Language-based Critiquing. arXiv:cs.CL/2005.11067

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.

Li Chen and Pearl Pu. 2012. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction* 22, 1-2 (2012), 125–150. https://doi.org/10.1007/s11257-011-9108-6

Boi Faltings, Pearl Pu, Marc Torrens, and Paolo Viappiani. 2004. Designing Example-Critiquing Interaction. In *Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI '04)*. Association for Computing Machinery, New York, NY, USA, 22–29. https://doi.org/10.1145/964442.964449

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.

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.

Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*. Association for Computing Machinery, New York, NY, USA, 133–142. https://doi.org/10.1145/775047.775067

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.

Kai Luo, Scott Sanner, Ga Wu, Hanze Li, and Hojin Yang. 2020a. Latent Linear Critiquing for Conversational Recommender Systems. In *Proceedings of the 29th International Conference on the World Wide Web (WWW-20)*. Taipei, Taiwan.

Kai Luo, Hojin Yang, Ga Wu, and Scott Sanner. 2020b. Deep Critiquing for VAE-Based Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1269–1278. https://doi.org/10.1145/3397271.3401091

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.

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.

James Reilly, Kevin McCarthy, Lorraine Mcginty, and Barry Smyth. 2004a. 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

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

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

Noveen Sachdeva and Julian McAuley. 2020. How Useful are Reviews for Recommendation? A Critical Review and Potential Improvements. arXiv:cs.IR/2005.12210

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.

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.

Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. 2018. Preference Elicitation as an Optimization Problem. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 172–180. https://doi.org/10.1145/3240323.3240352

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.

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

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.