Self-Supervised Bot Play for Conversational Recommendation with Justifications

Shuyang Li  
UC San Diego  
shl008@ucsd.edu

Bodhisattwa Prasad Majumder  
UC San Diego  
bmajumde@ucsd.edu

Julian McAuley  
UC San Diego  
jmcauley@ucsd.edu

Figure 1: Conversational critiquing workflow. The system scores candidate items and generates a justification for the top item. If the user rejects the suggestion and critiques an aspect, the system uses the critique to update the latent user representation.

1 INTRODUCTION

Traditional recommender systems often return static recommendations, with no way for users to meaningfully express their preferences and feedback. However, interactivity and explainability can greatly affect a user’s trust of and willingness to use a recommender system [29, 35]. This is reflected in human conversations: experts justify their recommendations, customers critique suggestions, and both parties iterate through the conversation to arrive at a satisfactory item.

Early work on interactive recommender systems focused on iteratively presenting suggestions to the user based on simple “like” and “dislike” feedback on individual items [3, 11, 15]. Gradually, systems began to accommodate more fine-grained user feedback—critiquing fixed attributes of an item (e.g., its color or brand) [6]. Recent models for conversational critiquing incorporate user feedback on subjective aspects (e.g., taste and perception) [22, 23, 40]. However, such methods are trained using a next-item recommendation objective, and perform poorly when engaging with users over multiple turns.

Another approach lies in training dialog agents to interact with the user over multiple turns [39]. While such models are able to generate convincing dialog in a vacuum, they require large corpora of transcripts from crowd-sourced recommendation games [12, 20]. To create high-quality training dialogs, crowd-workers must be knowledgeable about many items in the target domain—this expertise requirement limits data collection to a few common domains like movies. Additionally, these dialog policies limit a user’s freedom to interact with the system by asking yes/no questions about specific item attributes.

We thus desire a conversational recommender system that mimics characteristics of human interactions not yet captured by existing systems:

(1) It can justify suggestions made to the user;
(2) It updates suggestions based on user feedback about subjective aspects; and
(3) It can be trained using review data that is easily harvestable from arbitrary new domains.

ABSTRACT

Conversational recommender systems offer the promise of interactive, engaging ways for users to find items they enjoy. We seek to improve conversational recommendation via three dimensions: 1) We aim to mimic a common mode of human interaction for recommendation: experts justify their suggestions, a seeker explains why they don’t like the item, and both parties iterate through the dialog to find a suitable item. 2) We leverage ideas from conversational critiquing to allow users to flexibly interact with natural language justifications by critiquing subjective aspects. 3) We adapt conversational recommendation to a wider range of domains where crowd-sourced ground truth dialogs are not available. We develop a new two-part framework for training conversational recommender systems. First, we train a recommender system to jointly suggest items and justify its reasoning with subjective aspects. We then fine-tune this model to incorporate iterative user feedback via self-supervised bot-play. Experiments on three real-world datasets demonstrate that our system can be applied to different recommendation models across diverse domains to achieve superior performance in conversational recommendation compared to state-of-the-art methods. We also evaluate our model on human users, showing that systems trained under our framework provide more useful, helpful, and knowledgeable recommendations in warm- and cold-start settings.

KEYWORDS

classification, recommender systems, critiquing

ACM Reference Format:

Shuyang Li, Bodhisattwa Prasad Majumder, and Julian McAuley. 2021. Self-Supervised Bot Play for Conversational Recommendation with Justifications. In arXiv, 11 pages.

This paper is published under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 International (CC-BY-NC-ND 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

arXiv, 2021, USA  
© 2021
To accomplish these goals, we present a two-part framework to train conversational recommender systems. Ours is the first framework, to our knowledge, that allows training of conversational recommender systems for multi-turn settings without the need for supervised dialog examples. First, using a next-item recommendation task we learn to encode historical user preferences and generate justifications for our suggestions via sets of subjective aspects. We then fine-tune our trained model via multiple turns of bot-play in a recommendation game based on user reviews and simulated critiques. We apply our framework to two base recommender systems (PLRec [32] and BPR [30]), and evaluate the resulting PLRec-Bot and BPR-Bot models on three large real-world recommendation datasets containing user reviews. Our method reaches the target item with a higher success rate than state-of-the-art methods, and takes fewer turns to do so, on average. We also conduct a study with real users, showing that it can effectively help users find desired items in real time, even in a cold-start setting.

We summarize our main contributions as follows: 1) We present a framework for training conversational recommender systems using bot-play on historical user reviews, without the need for large collections of human dialogues; 2) We apply our framework to two popular recommendation models (BPR-Bot and PLRec-Bot), with each showing superior or competitive performance in comparison to state-of-the-art recommendation and critiquing methods; 3) We demonstrate that our framework can be effectively combined with query refinement techniques to quickly suggest desired items.

2 RELATED WORK

2.1 Recommendation with Justification

Users prefer recommendations that they perceive to be transparent or justified [33, 35]. Some early recommender systems presented the objective attributes of suggested items to users [34, 37], but did not attempt to personalize the justifications. Another line of work considered the problem of generating natural language explanations of recommendations. McAuley et al. [24] extract key aspects from the text of user reviews using topic extraction. Such justifications have been expanded into full sentences based on aspects of interest—constructed via template-filling [42] or recurrent language models [26]. Due to the unstructured nature of these justifications, however, sentence-level justifications have not been used for iteratively refining recommendations. In this work, justifications take the form of specific aspects that a user is interested in (e.g. that a song is poetic) [2, 24, 40]. In Section 4.2 we describe how we extract such aspects from user reviews in large recommendation datasets.

2.2 Conversational Recommendation

Users often seek to make informed decisions around consumption, and controllability of a recommendation system is linked to improved user satisfaction [27]. We thus turn to conversational recommendation as a way to iteratively engage with a user to learn their preferences with the goal of recommending a suitable item [5]. We view recommenders as domain experts engaging with human customers, able to elicit user preferences and requirements and suggest appropriate items in the course of the conversation.

In early interactive recommender systems, users were only able to give binary “like” and “dislike” feedback without further explanation [15]. One line of research used such feedback to refine the search space for retrieving desired images from the web [11]. Biswas et al. [3] extended this approach to interactive product search. More recently, multi-armed bandit approaches to conversational recommendation [43, 45] leverage exploration-exploitation algorithms to maximize the information learned from feedback at each turn.

Another line of work treats conversational recommendation as a form of task-oriented dialog where users express opinions about specific aspects of an item. At each turn, the user is either a) asked if they prefer a specified aspect; or b) recommended an item [7]. Self-supervised bot-play has been explored as a way to train such conversational dialog agents [12, 20], but such approaches require Wizard-of-Oz style data [8] with humans playing the role of expert and seeker. The quality of such dialog data depends heavily on the domain knowledge and competence of crowd-workers, which makes it unsuitable for complex domains. Zhang et al. [41] uses templated dialog forms and trains a model to ask about aspects that are most informative of the user’s preferences. However, this forces the user to answer yes/no questions and restricts their flexibility when giving feedback. Instead, we explore conversational critiquing, where a user is presented with items and justifications, and is able to give feedback regarding any aspect in the justification.
2.3 Conversational Critiquing

Critiquing systems aim to help users incrementally construct their preferences in a way that mimics how humans refine their preferences and constraints depending on conversation context [36]. Early critiquing methods relied on constraint-based programming to iteratively shrink the search space of items as users provided more critiques [4].

More recently, Wu et al. [40] introduced a critiquing model with justifications via a list of natural language aspects mined from user reviews. In this setting, users are able to interact with any aspect present in the generated justification. Antognini et al. [2] generate a single sentence of explanation alongside the set of aspects, but still require users to interact with the aspect set. Luo et al. [23] use a variational auto-encoder (VAE) [14] in place of the collaborative filtering model, learning a bi-directional mapping function between user latent representations and aspects they have expressed in reviews. Such models can generate high precision justifications but have shown poor multi-turn recommendation performance.

Latent Linear Critiquing (LLC) methods do not generate justifications and instead allow users to critique any aspect from the vocabulary [18, 22]. After training a matrix factorization model to predict ratings, these models then learn a linear regressor to recover user embeddings from their historical aspect usage frequency. A linear programming problem is then solved to weight a user’s vocabulary [18, 22]. After training a matrix factorization model to cover user embeddings from their historical aspect usage frequency.

Figure 2: In (latent) conversational critiquing, user feedback about aspects \( (c^0, c^1) \) modifies our prior latent user preference vector \( y_u^0 \) to bring it closer to the target item embedding.

3 MODEL

Our model consists of three sections, as seen in Figure 3:

1. A matrix factorization recommender model \( M_{\text{rec}} \) that learns to embed users and items in an \( h \)-dimensional latent space;
2. A justification head \( M_{\text{just}} \) that predicts the aspects of an item toward which the user holds preferences;
3. A critiquing function \( f_{\text{crit}} \) that modifies a user’s preference embedding based on user feedback about specific aspects.

Our model supports multi-turn critiquing as shown in Figure 2. At each turn of a conversation, a user may provide explicit feedback about aspects they dislike about the current set of recommendations in the form of a critique \( (c^t) \). The critiquing function \( f_{\text{crit}} \) then uses this critique to modify our latent user representation \( y_u \) in order to bring it closer to the user’s target item.

3.1 Base Recommender System

Our method can be applied to any recommender system \( M_{\text{rec}} \) that learns user and item representations. We demonstrate its effectiveness using two popular methods based on matrix factorization and linear recommendation.

Bayesian Personalized Ranking (BPR) [30] is a matrix factorization recommender system that seeks to decompose the interaction matrix \( R \in \mathbb{R}^{|U| \times |I|} \) into user and item representations [16]. BPR optimizes a ranked list of items given implicit feedback (a set of items with which a user has recorded a binary interaction).

We learn \( h \)-dimensional user and item embeddings \( y_u^{\text{MF}}, y_i^{\text{MF}} \), computing the score via the inner product: \( \hat{x}_{u,i} = (y_u^{\text{MF}}, y_i^{\text{MF}})^T \). At training time, the model is given a user \( u \), observed item \( i \in I_u \), and unobserved item \( j \in I_u \). We maximize the likelihood that the user prefers the observed item \( i \) to the unobserved item \( j \):

\[
L_R = P(i >_u j | \Theta) = \sigma(\hat{x}_{u,i} - \hat{x}_{u,j})
\]

where \( \sigma \) represents the sigmoid function \( \frac{1}{1 + e^{-x}} \).

Projected Linear Recommendation (PLRec) is an SVD-based method to learn low-rank representations for users and items via linear regression [32]. The PLRec objective minimizes the following:

\[
\arg\min_W \sum_u \| r_u - r_u V W^T \|_2^2 + \Omega(W)
\]

where \( V \) is a fixed matrix obtained by taking a low-rank SVD approximation of \( R \) such that \( R = U \Sigma V^T \), and \( W \) is a learned embedding matrix. We thus obtain an \( h \)-dimensional user embedding \( y_u^{\text{MF}} = r_u V \) and \( h \)-dimensional item embedding \( y_i^{\text{MF}} = W_i \).

3.2 Generating Justifications

Our justification model \( M_{\text{just}} \) consists of an aspect prediction head: a fully connected network with two \( h \)-dimensional hidden layers
We posit that the user’s latent representation can be partially ex-
where During training, the aspect encoder takes in the user’s aspect his-
3.3 Encoding Aspects

Algorithm 1: Bot play framework for fine-tuning conversa-
tional recommenders.

| Recommendation and Justification models $M_{rec}, M_{just}$; Critique model $M_{seeker}$; for each user $u$ do for goal item $g \in I_u^*$ (Evaluation set) do initialize loss $\mathcal{L}$; initialize $\gamma_u^0$ from $M_{rec}$; for turn $t \in \text{range}(1,T)$ do compute scores $\hat{x}_{u,i} = M_{rec}(\gamma_u^t, i) \forall i \in I$; $\mathcal{L} \leftarrow \mathcal{L} + \delta \cdot \mathcal{L}_{CE}(g, \hat{x}_{u,i})$; recommend item $\hat{t} = \text{arg max}_i \hat{x}_{u,i}$; if $\hat{t} = g$ then break session with success; generate justification $\hat{k}_{u,i} = M_{just}(\gamma_u^t, \hat{t})$; $M_{seeker}$ critiques justification: $c_u^t$; $\gamma_u^{t+1} \leftarrow f_{crit}(\gamma_u^t, c_u^t)$; end end

that predicts a score $s_{u,i,a}$ for each aspect $a$. This model takes as input the sum of the learned user and item embeddings $(y_u, y_i)$.

At training time, we incorporate an aspect prediction loss $L_A$ by computing the binary cross entropy (BCE) for each aspect:

$$L_A = - \frac{1}{|A|} \sum_{a=0}^{|A|} k_{u,i,a}^t \cdot \log p_{u,i,a} + (1 - k_{u,i,a}^t) \cdot \log(1 - p_{u,i,a}) \quad (3)$$

where $p_{u,i,a} = \sigma(s_{u,i,a})$ represents the likelihood of user $u$ caring about aspect $a$ in context of item $i$. At inference time, we again compute the likelihood for each aspect $(p_{u,i,a} = \sigma(s_{u,i,a}))$ and sample from the Bernoulli distribution with $p_{u,i,a}$ to determine which aspects $a$ appear in the justification.

3.3 Encoding Aspects

We posit that the user’s latent representation can be partially explained by their written reviews. Thus, we jointly learn an aspect encoder $M_{AE}$ alongside our recommendation model. This takes the form of a linear projection from the aspect space to the user preference space: $M_{AE}(c_u^t) = W^T c_u^t + b$, where $c_u^t \in \mathbb{Z}^{|K|}$ is the critique vector representing the strength of a user’s preference for each aspect. We then fuse this aspect encoding with the latent user embedding from $M_{rec}$ to form the final user preference vector $y_u$; $y_u = f(y_u^M, M_{AE}(c_u^t))$

For the BPR-based model, we use $f(a,b) = a + b$ as a fusion function, and for the PLRec-based model, we use $f(a,b) = \frac{a \cdot b}{1 + \frac{a}{b}}$. During training, the aspect encoder takes in the user’s aspect history: $c_u^t = k_u^U$.

3.4 Training

To train our BPR-based model, we jointly optimize each component. Each training example consists of a user $u$, an observed item $i \in I_u^*$ that the user has interacted with, and an unobserved item $j \in I_u$ that the user has not rated. We predict scores for items $i$ and $j$:

$$x_{u,i} = (y_u, y_i) = (y_u^M + M_{AE}(k_u^U), y_i) \quad (4)$$

We first compute the BPR loss (see Section 3.1) with predicted scores $x_{u,i}$ and $x_{u,j}$. We add the aspect prediction loss, scaled by a constant $\lambda_{KP}$ to the ranking loss for our training objective: $\mathcal{L} = \lambda_{KP} \mathcal{L}_A - \mathcal{L}_R$. We find empirically that $\lambda_{KP} \in [0.5, 1.0]$ works well.

To train our PLRec-based model, we follow Luo et al. [22] and separately optimize $M_{rec}, M_{just},$ and $M_{AE}$. The user and item embeddings are learned via eq. (2). We solve the following linear regression problem to optimize $M_{AE}$:

$$\arg \min_{w,b} \sum_u \| y_u^M - M_{AE}(k_u^U) \|_2^2 + \Omega(W) \quad (5)$$

Finally, we optimize the aspect prediction (justification) loss $L_A$ to train the justification head.

3.5 Conversational Critiquing with Our Models

To perform conversational critiquing with a model trained using our framework, we adapt the latent critiquing formulation from Luo et al. [22], as shown in Figure 1. Each conversation with a user $u$ consists of multiple turns. At each turn $t$, the system assigns scores $\hat{x}_{u,i}^t$ for all candidate items $i$, and presents the user with the highest scoring item $\hat{i}$. The system also justifies its prediction with a set of predicted aspects $\hat{k}_{u,i}^t$. The user may either accept the recommended item (ending the conversation) or critique an aspect from the justification: $a \in \{a|\hat{k}_{u,i,a} = 1\}$.

Given a user critique, the system modifies the predicted scores for each item and presents the user with a new item and justification:

$$\hat{x}_{u,i}^{t+1} = M_{rec}(\gamma_u^{t+1}, i) \quad (6)$$

$$\hat{k}_{u,i}^{t+1} = M_{just}(\gamma_u^{t+1}, i) \quad (7)$$

$$\hat{y}_{u,i}^{t+1} \leftarrow f_{crit}(\gamma_u^{t+1}, c_{u,i}^{t+1}) \quad (8)$$

Effectively, a user critique modifies our prior for the user’s preferences; we then re-rank the items presented to the user.

At inference time, a user critique modifies our prior for the user’s preferences; we then re-rank the items presented to the user.

At inference time, a user critique modifies our prior for the user’s preferences; we then re-rank the items presented to the user.

3.6 Learning to Critique via Bot Play

We propose a framework for critiquing via bot play that simulates conversations when provided just a known set of user reviews. We first pre-train our expert model (recommender model, justification model, and aspect encoder). A seeker model $M_{seeker}$ is pre-trained via a simple user prior: when provided with a known target item
and justification, it selects the most popular aspect present in the justification \(k_{u,t}^g\) but not the target’s historical aspects \(k_{i,j}^t\) to critique.

For each training example (user \(u\) and a goal item they have reviewed \(g\)), we allow the expert and seeker models to converse with the goal of recommending the goal item. We fine-tune the expert by maximizing its reward (minimizing its loss) in the bot-play game (Algorithm 1). We end the dialog after the goal item is recommended or a maximum session length of \(T = 10\) turns is reached. We define the expert’s loss as the cross entropy loss of recommendation scores per turn:

\[
L_{\text{expert}} = \sum_{t=1}^{T} \delta^{t-1} \cdot L_{\text{CE}}(g, \hat{x}^t_{u,i})
\]

where \(\delta\) is a discount factor\(^1\) to encourage sequentially recommending the goal item at earlier turns. \(L_{\text{CE}}(g, \hat{x}^t_{u,i})\) is the cross entropy loss between predicted scores and the goal item:

\[
L_{\text{CE}}(g, \hat{x}^t_{u,i}) = -\sum_{i \in I} P(i) \log_Q(i); \quad Q(i) = \frac{e^{\hat{x}^t_{u,i}}}{\sum_i e^{\hat{x}^t_{u,i}}}
\]

where \(P(i)\) is 1 if \(g = i\) and 0 otherwise. As the cross-entropy loss is continuous, we optimize the reward for each conversation \((u,i)\).

4 EXPERIMENTAL SETTING

To train our initial model, we select hyperparameters via AUC on the validation set. We select hyperparameters for bot-play fine-tuning by evaluating the success rate at 1 (SR@1) on the validation set. We train each model once, with three evaluation runs per experimental setting. For baseline models, we re-used the authors’ code. We include additional training details in the supplementary materials. We will make our code available upon publication.

4.1 Datasets

We evaluate the quantitative performance of our model using three real-world, publicly available recommendation datasets: Goodreads Fantasy (Books) [38], BeerAdvocate (Beer) [24], and Amazon CDs & Vinyl (Music) [10, 25]—each with over 100K reviews and ratings. We keep only reviews with positive ratings, setting thresholds of \(t > 4.0\) for Beer and Music and \(t > 3.5\) for Books. We partition each dataset into 50% training, 20% validation, and 30% test splits. Dataset statistics are shown in Table 3.

4.2 Aspect Extraction

Our datasets do not contain pre-existing aspects, so we follow the pipeline of [40] to extract subjective aspects from user reviews:

\(^1\)We use a discount factor of \(\delta = 0.9\)

Table 3: Descriptive statistics of datasets, including average unique aspects expressed in reviews per item and user.

|       | Users | Items | Reviews | Asp. | Asp./Item | Asp./User |
|-------|-------|-------|---------|------|-----------|-----------|
| Books | 13,889| 7,649 | 654,975 | 75   | 27.0      | 25.0      |
| Beer  | 6,369 | 4,000 | 933,524 | 75   | 60.2      | 54.6      |
| Music | 5,635 | 4,352 | 119,081 | 80   | 20.0      | 16.5      |

4.3 Multi-Step Critiquing

Following prior work on conversational critiquing [18, 22], we simulate multi-step recommendation dialogs to assess model performance. We randomly sample 500 user-item interactions from the test set to conduct user simulations following Algorithm 2 for each user \(u\) and goal item \(g\). At each turn, we recommend an item \(\hat{I}\) to the user alongside a set of aspects \(\hat{k}_{u,t}^g\). If the goal item is not recommended, the user will critique an aspect \(\hat{k}_{u,t}^g\) from \(\hat{k}_{u,t}^g\) to improve the recommendation.

To evaluate how our models behave with different user behaviors, we simulate each observation with three different critique selection strategies [18]:

- **Random**: We assume the user randomly chooses an aspect.
  This assumes no prior knowledge on the part of the user.
- **Pop**: We assume the user selects the most popular aspect used across all training reviews.
- **Diff**: We assume the user selects the aspect that deviates most from the goal item reviews. In simulations, we select

Algorithm 2: User Simulation Evaluation

Recommendation and Justification models \(M_{\text{rec}}, M_{\text{just}}\):

Critique fusion function \(f_{\text{crit}}\):

for each user \(u\) do

for goal item \(g \in I^*_u\) (Evaluation set) do

initialize \(y_{u,t}\) from \(M_{\text{rec}}\);

for turn \(t \in \text{range}(1, T)\) do

compute scores \(x^t_{u,i} = M_{\text{rec}}(y^t_u, i)\) \(\forall i \in I\);

recommend item \(\hat{I} = \arg \max_i x^t_{u,i};\)

generate justification \(k^t_{u,i} = M_{\text{just}}(y^t_u, y^t_p)\);

if \(I^* = g\) then

break session with success;

end

user critiques justification: \(c^t\);

\(y^{t+1} = f_{\text{crit}}(y^t_u, c^t_u)\);

end

end

end

return average success rate & length

(1) Extract lists of high-frequency unigrams and bigrams (nouns and adjective phrases only) from all user reviews;

(2) Prune the bigram keyphrase list using a Pointwise Mutual Information (PMI) threshold, ensuring aspects are statistically unlikely to have randomly co-occurred;

(3) Represent reviews as sparse binary vectors indicating whether each aspect was expressed in the review.

Aspects describe a wide range of qualities; for beers, users commonly describe the malt (e.g. roasted) and taste (e.g. citrus). For music, aspects range from perceived genres (e.g. techno) to emotions (e.g. soulful). Users describe books by reacting to character descriptions (e.g. strong female) and settings (e.g. realistic).
the aspect with the largest frequency differential between the goal item and current item:

$$\text{arg max}_a (k^{I}_{p,a} - k^{I}_{g,a})$$

In all critiquing settings, a user may not critique the same aspect multiple times in a session, and any recommended items are removed from consideration in the following turns.

### 4.4 Candidate Algorithms

As our method can be applied to any base recommender system $M_{rec}$, we apply our framework to train models based on BPR and PLRec (see Section 3.1)—**BPR-Bot** and **PLRec-Bot**, respectively.

We assess Latent Linear Critiquing (LLC) baselines, which embed critique vectors $c^t_u$ in the same $h$-dimensional space as the latent user representation $\gamma_u$. $f_{crit}$ is defined as a weighted sum of the embedding for each critiqued aspect, alongside the original user preference vector. **UAC** [22] averages the initial user embedding and all critiqued aspect embeddings. **BAC** [22] first averages critiqued aspects, and then averages the result with the initial user embedding. **LLC-Score** [22] learns the weights via a linear program maximizing the posterior rating differences between items containing critiqued aspects and those without. Instead of directly optimizing the scoring margin, **LLC-Rank** [18] minimizes the number of ranking violations. These models cannot generate justifications; we binarize the historical aspect frequency vector for the item $(k^{I}_{p,a})$ as a justification at each turn. We compare against these models to evaluate whether generating personalized justifications can improve critiquing.

We also compare against a state-of-the-art variational conversational recommender, **CE-VAE** [23]—an improvement on the Wu et al. [40] justified critiquing model—which jointly learns to recommend and justify. CE-VAE learns a VAE with a bidirectional mapping between critique vectors and the user latent preference space. We compare our models to CE-VAE to assess how justification quality impacts multi-turn critiquing performance.

### 5 EXPERIMENTS

**RQ1: Can our framework enable multi-step critiquing?** To measure multi-step critiquing performance, we assess the average success rate and session length following Luo et al. [22]. Success rate measures the percentage of sessions in which the target item reaches rank $N$, and session length measures the average length of sessions with a limit of 10 iterations. Success rates and session lengths for each dataset and user behavior model are shown in Figure 4 and Figure 5, respectively.

Our models are fine-tuned via bot-play with a seeker model that assumes one particular user behavior: popularity-based critique selection. As such, we expect it to perform better in the Pop user setting. However, BPR-Bot and PLRec-Bot succeeds at a higher rate in fewer turns than baselines under all user settings—including random aspect critiquing, which assumes no prior on user behavior.

**Variational Baseline.** Despite its strong first-turn recommendation performance and high-fidelity justifications, CE-VAE is outperformed by our models in all nine settings across all metrics. This supports our observation that the training method to learn...
with a limited set of broad requirements (e.g. when buying a car, (UAC, BAC, LLC-Score, and LLC-Rank) perform poorly on multi-via bot-play. For clarity, we display only results using the Pop user conversation knowing their full preference set \[ 28 \], they often start confers a 3-6% improvement in success rate for various N. This and Diff user models. In domains with relatively high aspect oc-

quickly reach significantly higher success rates.

model can be combined with constraint-based query refinement to

an initial set of user requirements—while users rarely enter a con-

from constraint-based filtering as well as starting the session from

geometric critiquing systems may benefit

demonstrates that we can effectively train conversational recom-
mender systems using our bot-play framework using domains with

rich user reviews in lieu of crowd-sourced dialog transcripts.

In domains with more sparse coverage of subjective aspects

(\text{i.e. Music}), we observe lower improvement when using bot-play. Here, our model may not encounter sufficient examples of rare aspects being critiqued. In future work, we will explore methods to add noise to our user model to ensure that the bot-play process encounters more rare aspects. We will also investigate additional losses for bot-play, including ranking losses instead of cross entropy.

We confirm that our method is model-agnostic, as it improves conversational recommendation success rates for both the matrix factorization-based (BPR) and linear (PLRec) recommender systems. We also observe that models with a higher latent dimensionality \( h \in [50, 400] \) for PLRec-Bot vs. \( h = 10 \) for BPR-Bot) benefit more from bot-play, suggesting that our method effectively learns to navigate complex user preference spaces.

**RQ3: Can our models be effectively combined with query refinement?** So far, we have assumed that users provide soft feedback via critiques: even if a user has critiqued aspect \( a \) during a session, future suggested items may still contain aspect \( a \). This assumption holds for some aspects: for example, even if previous users mentioned that a song was dispassionate, a user may find it emotional and enjoyable. However, in a real-world setting that user may reject the suggestion after reading the reviews. Thus, we experiment with treating critiques as hard feedback: if a user critiques some aspect \( a \), we prune all candidate items whose reviews have not expressed aspect \( a \) in this session, future suggested items may still contain aspect \( a \).

We compare three models in this setting, with the turn-0 ranked list of candidate items initialized from BPR-Bot.

The **Query** baseline model suggests one item per turn and asks the user whether they like aspect \( a \). If the user answers yes, we prune all candidate items whose reviews have not expressed aspect \( a \): \( t^{t+1} \leftarrow \{ i \mid i \in t^t \&|k_{t,i}^a| = 1 \} \). Otherwise, we prune all candidates whose reviews have expressed the aspect: \( t^{t+1} \leftarrow \{ i \mid i \in t^t \&|k_{t,i}^a| = 0 \} \). At each turn, we pick the aspect that most evenly divides the remaining candidate items: \( \text{arg min}_{a} |||t^t|| - |t^t||a|| \). Effectively, we perform binary search over our candidate space, and expect to find the target item within \( \log |t| \) turns.

In the **Filter** model, we suggest an item alongside a generated justification per turn. When a user critiques aspect \( a \), we prune candidate items whose reviews have expressed aspect \( a \): \( t^{t+1} \leftarrow \{ i \mid i \in t^t \&|k_{t,i}^a| = 0 \} \). We extend this model via our learned critiquing function \( f_{\text{crit}} \) to further modify the user preference vector and
Table 4: Conversation-level human evaluation via ACUTE-EVAL. Win (W) and Loss (L) percentages are reported while ties are not. All results statistically significant with \( p < 0.05 \).

| BPR-Bot   | Useful | Informative | Knowledgeable | Adaptive | vs  |
|-----------|--------|-------------|---------------|----------|-----|
|           | W L    | W L         | W L           | W L      |     |
| Ablation  | 78 10  | 73 11       | 68 15         | 85 5     |     |
| CE-VAE    | 83 9   | 74 10       | 63 16         | 81 8     |     |

| PLRec-Bot | Useful | Informative | Knowledgeable | Adaptive | vs  |
|-----------|--------|-------------|---------------|----------|-----|
|           | W L    | W L         | W L           | W L      |     |
| Ablation  | 86 5   | 78 7        | 74 8          | 81 9     |     |
| CE-VAE    | 87 7   | 79 11       | 77 12         | 83 10    |     |

Each sample is evaluated by three annotators. We observe substantial [17] inter-annotator agreement, with Fleiss Kappa [9] of 0.67, 0.79, 0.73, and 0.60 for the usefulness, informativeness, knowledgeable, and adaptiveness criteria, respectively.

BPR-Bot and PLRec-Bot are judged to be significantly more informative and knowledgeable compared to ablative models and CE-VAE, showing that our justification module accurately predicts aspects of a recommended item. We design the usefulness and adaptiveness criteria to capture how our framework aids the user in achieving their conversational goal (i.e. recommending the most relevant item within a minimum number of turns). Compared to the alternatives, models trained under our bot-play framework are judged to be more useful and adapt their recommendations in a manner more consistent with critiques.

Our framework allows us to train conversational agents that are useful and engaging for human users: evaluators overwhelmingly judged the models trained via bot-play to be more useful, informative, knowledgeable, and adaptive compared to CE-VAE and ablated variants.

**Cold-Start User Study** We conduct a user study using items and reviews from the Books dataset to evaluate our model’s ability to provide useful conversational recommendations in real-time. We recruited 32 real human users to interact with our BPR-Bot recommender and another 32 to interact with the ablative model (no conversational fine-tuning). As evaluators do not correspond to users in our training data, we initialize each conversation with the average of all learned latent user representations.

At each turn, the user is presented with the three top-ranked items and their justifications (list of aspects), and is allowed to critique multiple aspects. On average, users critiqued two aspects per turn—which suggests that when training conversational models we should assume multiple critiques at each turn.

We evaluate our systems following Li et al. [19]: at each turn, we ask our users if the generated justifications are informative, useful in helping to make a decision, and whether our system adapted its suggestions in response to the user’s feedback. We provide four options for each question: yes, weak-yes, weak-no, and no, mapping these values to a score between 0 and 1 [13]. We display the normalized aggregated score for each question in Table 5. We find that BPR-Bot significantly out-scores the ablative model in all three metrics (\( p < 0.01 \)), showing that fine-tuning our model on a bot-play framework instills a stronger ability to respond to techniques and provide meaningful justifications—even for unseen users.

At the end of a conversation, we additionally ask the user how frequently (if at all) they would choose to engage with our conversational agent in their daily life. 69% of users indicated they would “often” or “always” use BPR-Bot to find books, compared to 41% of users for the ablative model. We thus find that fine-tuning
our model via bot-play also makes it significantly (p < 0.05) more useful for new users.

7 CONCLUSION
In this work, we aim to develop conversational agents for recommendation that engage with users following common modes of human dialog: justifying why suggestions were made and incorporating feedback about certain aspects of an item to provide better recommendations at the next turn. We present a framework for training conversational recommenders in this modality via self-supervised bot-play. Our framework is model-agnostic and allows conversational recommenders to be trained on any domain with review data. We use two popular underlying recommender systems to train the BPR-Bot and PLRec-Bot conversational agents using our framework, demonstrating quantitatively on three datasets that our models: 1) offer superior multi-turn recommendation performance compared to current state-of-the-art methods; 2) can be effectively combined with query refinement techniques to quickly converge on suitable items; and 3) can iteratively refine suggestions in real-time, as shown in user studies. In future work, we aim to adapt our framework to natural language critiques (i.e. complete utterances), allowing users to freely express their feedback in a less restrictive way.

REFERENCES

[1] Diego Antognini and Boi Faltings. 2021. Fast Multi-Step Critiquing for VAE-Based Recommender Systems. CoRR abs/2105.00774 (2021). arXiv:2105.00774

[2] Diego Antognini, Claudio Musat, and Boi Faltings. 2020. Interacting with Explaining through Critiquing. CoRR abs/2005.11067 (2020). arXiv:2005.11067

[3] Ari Biwas, Thai T. Pham, Michael Vogelshong, Benjamin Snyder, and Houssam Nassif. 2019. Seeker: Real-Time Interactive Search. In KDD. https://doi.org/10.1145/3292500.3307373

[4] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. 1996. Knowledge-Based Navigation of Complex Information Spaces. In AAAI.

[5] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. 1997. The FindMe Approach to Assisted Browsing. IEEE Expert 12, 4 (1997), 32–40. https://doi.org/10.1109/64.608186

[6] Li Chen and Pearl Pu. 2012. Critiquing-based recommenders: survey and emerging trends. UMUIA (2012).

[7] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems. In KDD. https://doi.org/10.1145/2939672.2939746

[8] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. 1993. Wizard of Oz studies: a case study with SetFusion. In CHI. https://doi.org/10.1145/1639714.1639777

[9] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations at the next turn. In RecSys. https://doi.org/10.1145/1639721.3340733

[10] Raymond Li, Scott Tanner, Kai Luo, and Ga Wu. 2020. A Ranking Optimization Approach to Latent Linear Critiquing for Conversational Recommender Systems. In RecSys. https://doi.org/10.1145/3383313.3412240

[11] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. Biometrics 33, 1 (1977), 159–174. http://www.jstor.org/stable/2529320

[12] Xiangyu Jin and James C. French. 2003. Improving image retrieval effectiveness in a restrictive way.

[13] Xiyuan Liu, Haoming Jiang, Pengcheng He, Weiizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jianwei Han. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In ICLR.

[14] Julian J. McAuley, Scott Tanner, Ga Wu, Hanze Li, and Hoi-Jun Yang. 2020. Latent Linear Critiquing for Conversational Recommender Systems. In WWW. https://doi.org/10.1145/3364623.3368003

[15] Kai Luo, Hoi-Jun Yang, Ga Wu, and Scott Tanner. 2020. Deep Critiquing for VAE-Based Recommender Systems. In SIGIR. https://doi.org/10.1145/3397271.3401091

[16] Julian J. McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning Attributes and Attributes from Multi-aspect Reviews. In ICDM. https://doi.org/10.1109/ICDM.2012.110

[17] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substyles. In SIGIR. https://doi.org/10.1145/2766462.2767755

[18] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. 1993. Wizard of Oz studies: a case study with SetFusion. In CHI. https://doi.org/10.1145/1639714.1639777

[19] Noveen Sachdeva and Julian J. McAuley. 2020. How Useful are Reviews for Recommendation? A Critical Review and Potential Improvements. In SIGIR. https://doi.org/10.1145/3397271.3401281

[20] Nihat Tintarev and Judith Mashoff. 2011. Designing and Evaluating Explanations for Recommender Systems. In Recommender Systems Handbook. Springer, 479–510.

[21] Amos Tversky and Iram Simonson. 1993. Context-Dependent Preferences. Management Science 39, 10 (1993), 1179–1189.

[22] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagplanations: explaining recommendations using tags. In IJCAI. https://doi.org/10.1561/7100000061

[23] Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-turn Comparisons. arXiv:1909.03087 [cs.CL]

[24] Raymond Li, Samira Bibraini Kahu, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In NeurIPS.

[25] Xiyuan Jin and James C. French. 2003. Improving image retrieval effectiveness in a restrictive way.

[26] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations via Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.

[27] Jun Sun, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.

[28] Deniz Parra and Peter Bruslovsky. 2015. User-controllable personalization: a case study with setfusion. In Int. J. Hum. Comput. Stud. 78 (2015), 43–67. https://doi.org/10.1016/j.ijhcs.2015.01.007

[29] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2019. Seeker: Real-Time Interactive Search. In EMNLP-IJCNLP.

[30] Dongyong Kong, Ansa-Maria Camburu, Leonard Salewski, Cornelius Emde, Virgie Do, Zeynep Akata, and Thomas Lukasiewicz. 2021. e-VIL: A Dataset and Benchmark for Natural Language Explanations in Vision-Language Tasks. CoRR abs/2105.03761 (2021). arXiv:2105.03761

[31] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In ICLR. https://doi.org/10.1101/ICLR.2002.1047418

[32] ByoungChul Ko and Hyeyum Byun. 2002. Probabilistic Neural Networks Supporting Multi-Class Relevance Feedback in Region-Based Image Retrieval. In IJCR. https://doi.org/10.1101/ICPR.2002.1047418

[33] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[34] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In SIGIR. https://doi.org/10.1145/3397271.3401181

[35] Jun Sun, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.

[36] Jun Sun, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.

[37] Jun Sun, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.

[38] Jun Sun, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.

[39] Jun Sun, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In EMNLP.
A ADDITIONAL TRAINING DETAILS

All experiments were conducted on a machine with a 2.2GHz 40-core CPU, 132GB memory and one RTX 2080Ti GPU. We use PyTorch version 1.4.0 and optimize our models using the Rectified Adam [21] optimizer. Best hyperparameters for each base recommender system model are shown in Table 7.

B TIME COMPLEXITY

In Table 6, we report the mean and standard error of time taken per turn for LLC-Score, CE-VAE, BPR-Bot, and PLRec-Bot. As baseline code does not leverage the GPU, we also critique with PLRec-Bot and BPR-Bot on the CPU only. We observe LLC-Score and PLRec-Bot to be an order of magnitude slower per critiquing cycle compared to CE-VAE and BPR-Bot.

BPR-Bot shows acceptable latency for real-world applications (sub-10 ms), and we observe empirically in our cold-start user study (Section 6) that we can host BPR-Bot as a real-time recommendation service. Time trials were conducted with batch size of 1; production throughput can be improved further with parallel processing. Each model executes using a different framework (NumPy for LLC-Score, TensorFlow for CE-VAE, and PyTorch for PLRec-Bot/BPR-Bot), which may contribute to differences in inference speed.

C USER EVALUATION

An image of the interface used for our cold-start user study (Section 6) is shown in Figure 8.

### Table 6: Mean and standard error of wall-clock time (ms) per turn of critiquing for linear (LLC-Score) and variational (CE-VAE) baselines vs. our models (BPR-Bot, BPR-PLRec)

|          | LLC-Score | CE-VAE  | BPR-Bot | PLRec-Bot |
|----------|-----------|---------|---------|-----------|
| Books    | 40.64 ± 20.46 | 4.61 ± 1.16 | 2.70 ± 3.95 | 48.84 ± 14.08 |
| Beer     | 15.94 ± 14.52 | 3.26 ± 1.18 | 2.54 ± 2.36 | 49.43 ± 14.81 |
| Music    | 42.21 ± 21.04 | 3.36 ± 1.37 | 2.25 ± 0.62 | 6.80 ± 7.53 |

Figure 8: User interface for user study, with turn-level feedback prompts and an example of a critiqued aspect ("Battle")
Table 7: Best hyperparameter settings for each base recommendation model. UAC, BAC, LLC-Score, LLC-Rank models use PLRec as a base model. BPR-Bot uses BPR as a base model.

| Dataset | Model  | \( h \) | LR    | \( \lambda_{L2} \) | \( \lambda_{KP} \) | \( \lambda_c \) | \( \beta \) | Epoch | Dropout |
|---------|--------|---------|-------|-----------------|-----------------|--------------|---------|-------|---------|
| Books   | BPR    | 10      | 0.001 | 0.01            | 0.5             | -            | -       | 200   | -       |
|         | PLRec  | 50      | -     | 80              | -               | -            | -       | 10    | -       |
|         | CE-VAE | 100     | 0.0001| 0.0001          | 0.01            | 0.01         | 0.001   | 300   | 0.5     |
| Beer    | BPR    | 10      | 0.001 | 0.01            | 0.5             | -            | -       | 200   | -       |
|         | PLRec  | 50      | -     | 80              | -               | -            | -       | 10    | -       |
|         | CE-VAE | 100     | 0.0001| 0.0001          | 0.01            | 0.01         | 0.001   | 300   | 0.5     |
| Music   | BPR    | 10      | 0.01  | 0.1             | 1.0             | -            | -       | 200   | -       |
|         | PLRec  | 400     | -     | 1000            | -               | -            | -       | 10    | -       |
|         | CE-VAE | 200     | 0.0001| 0.0001          | 0.001           | 0.001        | 0.0001  | 600   | 0.5     |