# K-nearest Neighbors based Latent Critiquing for Conversational Recommender Systems

ECE499Y1 Interim Report

Student Name: Hanze Li

Student Number: 1002526493

Supervisor: Professor Scott Sanner

An undergraduate thesis submitted in fulfillment of the requirements
for the degree of Bachelor of Applied Science

In the
Data Driven Decision Making Lab
Department of Mechanical and Industrial Engineering

Engineering Science, Robotics Option
University of Toronto
January 20th, 2020

# Contents

# List of Figures



# List of Tables

## Abstract

Critiquing is a method for conversational recommendation that iteratively adapts recommendations in response to user preference feedback. Concretely, a user in simulation is iteratively provided with item recommendation and attribute description for the items; the user may either accept the recommendation, or choose to critique/boost a keyphrase to generate a new recommendation. Historical critiquing methods that based on latent embeddings largely used implicit method for updating recommendation results such as autoencoder model and projected linear recommendation and thus lacks interpretability and cannot precisely describe impact on critiques in recommendation process. In this thesis, we revisit the traditional User-based Collaborative Filtering with K-nearest Neighbours which makes personalized prediction. The method that we use leverages explicit user similarity as latent space for combining initial recommendation and critiques to update recommendation results, which makes it possible to trace both the source of explanation and impact of critiquing explicitly. A crucial research problem also arises here: how to weight the multi-step critiquing feedback to optimize recommendation given that each feedback is not necessarily of equal importance. To answer this question, we proposed various objects for optimization and evaluate them on Yelp open source dataset containing user reviews.

## 1 Introduction

Critiquing is a method for conversational recommendation that adapts recommendations in response to user preference feedback. For example, in unit critiquing, a user might critique a restaurant recommendation by requesting a restaurant to be more money-efficient and in compound critiquing, a user might further explore restaurants that have better ambiance and quick service than an initial recommendation. Further extensions such as incremental critiquing consider the cumulative effect of iterated critiquing interactions while experience-based methods attempt to collaboratively leverage critiquing interactions from multiple users.

Historical critiquing methods that based on latent embeddings largely used implicit method for updating recommendation results such as autoencoder model and projected linear recommendation and thus lacks interpretability and cannot precisely describe impact on critiques in recommendation process.In this thesis, we revisit the traditional User-based Collaborative Filtering with K-nearest

Neighbours which makes personalized prediction. In addition, we assume that item attributes are not explicitly known, but rather represented as keyphrases sourced from subjective user reviews. The method that we use leveraged explicit user similarity as latent space for combining initial recommendation and sequence of critiquings to update recommendation results, which makes it possible to trace both the source of explanation and impact of critiquing explicitly. We evaluate our proposed methods for K-nearest Neighbours based latent critiquing on Yelp open source datasets containing user reviews. In summary, this thesis provides a novel critiquing method for manipulating latent user embeddings and provides explicit and interpretable results for each critique.

# 2 Preliminaries

## 2.1 Notation

Notations are defined first for easier communication in the following sections.

- $R \in \mathbb{R}^{|I| \times |J|}$. User preference matrix. Entries $r_{i,j}$ are either 1 (preference observed) or 0 (preference not observed). $\mathbf{r}_i$ represents all feedback from user $i$, and $\mathbf{r}_{:,j}$ represents all user feedback for item $j$.

- $S \in \mathbb{R}^{|I| \times |K|}$. 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 $\mathrm{s}_i$ to represent $i$th user's keyphrase frequencies, and $\mathrm{s}_{:,k}$ to represent $k$th keyphrase's frequency across all users.

- $S' \in \mathbb{R}^{|J| \times |K|}$. This is the item-keyphrase matrix. Given item reviews written by all users from a corpus, we extract keyphrases that describe item attributes as shown in Table 1. This matrix contains items and term frequencies of keyphrases. We use $\mathrm{s}'_j$ to represent $j$th item's keyphrase frequencies, and $\mathrm{s}'_{:,k}$ to represent $k$th keyphrase's frequency across all items.

- $j^{-k} \in \{j | S'_{j,k} = 0, \forall j\}$. This item set represents items that do not contain the critiqued keyphrase $k$.

- $j^{+k} \in \{j | S'_{j,k} > 0, \forall j\}$. This item set represents items that contain the critiqued keyphrase $k$.

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

| Dataset | Reason Type | Keyphrases |
|---|---|---|
| Yelp | Category | Chinese, Thai, Italian, Mexican |
| | Food | Chicken, Beef, Fish, Pork, |
| | Drink | Tea, Beer, Coffee, Bubble Tea |
| | Ambiance | Vibe, Atmosphere, Service, Environment |
| | Price | Cheap, Pricy, Expensive |

## 2.2 User-based Collaborative Filtering with K Nearest Neighbour

**Collaborative Filtering (CF)** is a technique widely used in recommendation systems that makes automatic predictions about the interests of a user by collecting preferences from other users collaboratively. The underlying assumption here is that similar users identified by their preference history may share interests in items. **K Nearest Neighbour** method in CF takes k closest users in the feature space and collaboratively make predictions. To accomplish this, a user-similarity matrix is often constructed from the user-item preference matrix via various methods such as matrix multiplication and cosine similarity. The idea of user-similarity matrix was adopted and used as latent space for recommending new items for users.

## 2.3 Linear Recommendation

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

$$\underset{W}{\operatorname{argmin}} \sum_{i} ||\mathbf{r}_i - \mathbf{r}_i W^T||_2^2 + \Omega(W), \tag{1}$$

where $W$ represents the similarity matrix to train, and $\Omega$ is a regularization term. A well known linear recommender, *SLIM*, is introduced in the year of 2011, which optimizes the linear regression objective with zero-diagonal matrix constraint that prevent trivial solutions. Suvash et al. extended the SLIM algorithm by formally introducing alternative training schema that optimizing linear regression for each user, which further improves the scalability of the linear recommenders. This idea is being deployed for embedding user critiquing into the user similarity latent space.

## 2.4 Conversational Critiquing

In the **conversational critiquing** setting of this thesis, 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. An example of a single step conversational critiquing interaction from our experimentation are provided in Table 2.

Formally, critiquing at a single time step can be viewed as the process of embedding user critiquing $\mathbf{s}_i$ to latent space via linear transformation and produce modified prediction $\hat{\mathbf{r}}_i$ for user $i$.

$$\hat{\mathbf{r}}_i = f_m(\mathbf{r}_i, \tilde{\mathbf{s}}_i), \quad \textit{given} \quad \tilde{\mathbf{s}}_i = \psi(\mathbf{s}_i, \mathbf{c}_i), \tag{2}$$

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

## 2.5 Latent Linear Critiquing

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

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

$$\hat{r}_{i,j} = \langle \mathbf{z}_i, \mathbf{w}_j \rangle, \tag{3}$$

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

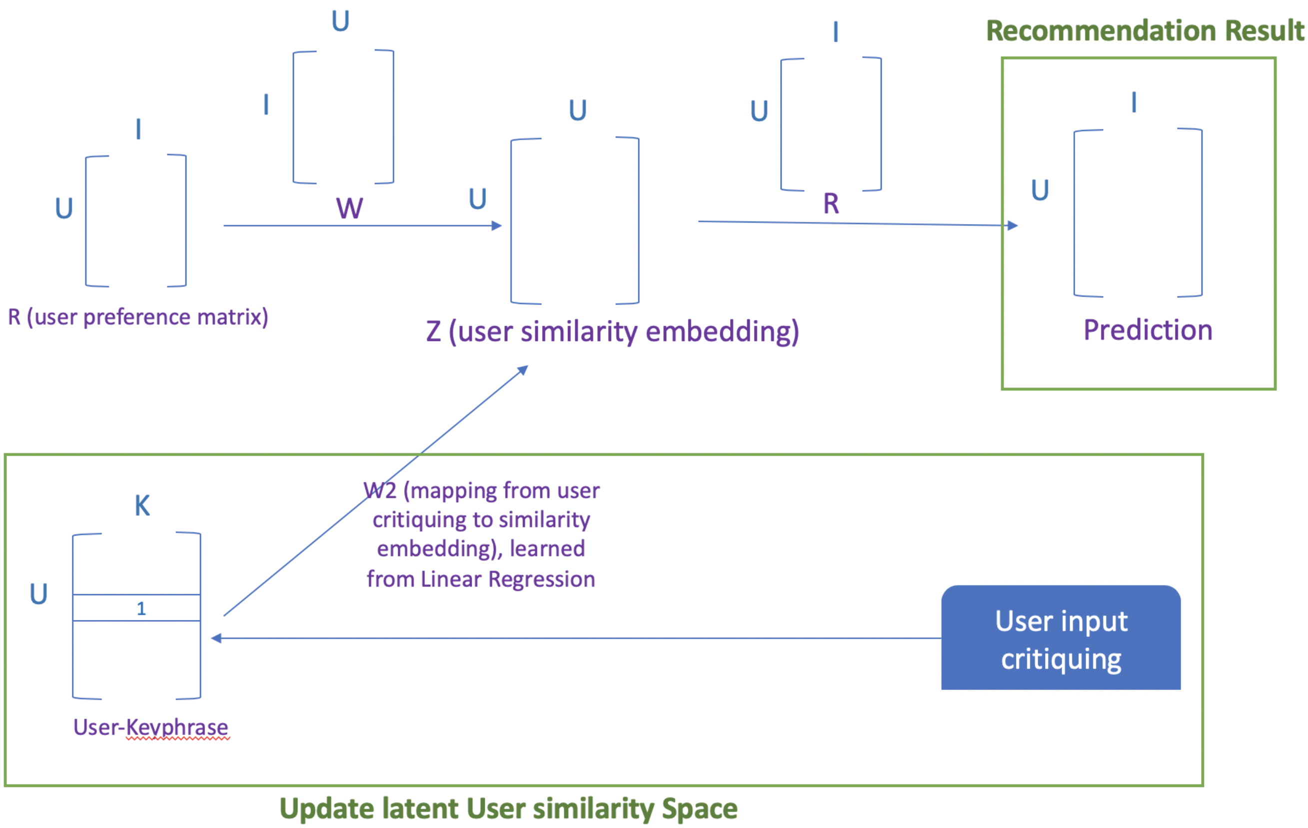


Figure 1: The structure of critiqued recommendation $f_m(\cdot)$. Historical user preferences $R$ and critiqued keyphrases $\tilde{\mathbf{s}}_i$ from user $i$ flow in at the left and are embedded into latent similarity spaces $Z$. These latent embeddings are merged with initial embedding and finally decoded into new item recommendations $\hat{\mathbf{r}}_i$ for user $i$ on the right that take into account their critiques.

With the user's critiques $\mathbf{c}_i$ in one-hot encoding form and the mapping between keyphrase and user latent representation learned in Equation (1), we are able to provide a latent representation of *all* critique:

$$\tilde{\mathbf{z}}_i^t = \mathbf{c}_i w_2^T + b, \tag{4}$$

where $\tilde{\mathbf{z}}_i^k$ represents latent representation of the critiqued keyphrase in similarity space, and $b$ is the bias term b. Notice that here, we can interpret the latent representation as how this critique affect user's similarity to other users and thus providing a meaningful and interpretation of impact.

Then under an assumption that critiques at each time step should be weighted equally (with the user preference), we can perform simple **Uniform Average Critiquing** for single step by defining

Table 2: User Case Study for single step critiquing on Yelp dataset.

| Target item | Initial recommended item | Initial Rank of target item | Keyphrase | Target item rank after critiquing |
|---|---|---|---|---|
| Wok&Roast Chinese BBQ | Khao San Road | 755 | Fried Rice | 77 |

the merging function as

$$\phi_\lambda(\mathbf{z}_i, \tilde{z}_i^t) = \lambda_0 \mathbf{z}_i + \lambda_1 \tilde{\mathbf{z}}_i^1 \tag{5}$$

where $\lambda_0$ and $\lambda_1$ are identical and sum to 1 (hence both equals to 0.5 in single step critiquing). K-nearest Neighbours method is then used for generating updated recommendations given the modified latent similarity embedding.

# 3 Dataset

We evaluate the proposed latent linear critiquing framework on one publicly available datasets: Yelp Review Dataset that contains more than 100,000 reviews and restaurant rating records.

## 3.1 Keyphrases Selection

Attributes associated with each restaurant in the Yelp Business data-set (such as restaurant category, restaurant take out, parking information) were given but in our case not enough for generating a dense user/item-keyphrase matrix and thus keyphrases were selected from subjective user reviews.

The following generic processing steps were followed to extract candidate keyphrases from the reviews to be used for explanation and critiquing:

1. Extract separate unigram and bigram lists of noun and adjective phrases from reviews of the entire dataset based on TF-IDF discussed above.

2. Prune the bigram keyphrase list using a Pointwise Mutual Information (PMI) threshold to ensure bigrams are statistically unlikely to have occurred at random.

3. Represent each review as a sparse 0-1 vector indicating whether each keyphrase occurred in the review.

Table 3: K-nearest Neighbours result (initial recommendation) with both implicit(binarized) and explicit user preference representation .

| res | R-precision | NDCG | MAP@5 | MAP@10 | MAP@50 | Precision@5 | Precision@10 | Precision@50 | Recall@5 | Recall@10 | Recall@50 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| explicit | 0.0721 | 0.1588 | 0.0927 | 0.0848 | 0.0638 | 0.0839 | 0.0732 | 0.0498 | 0.0499 | 0.0847 | 0.269 |
| implicit | 0.0526 | 0.1344 | 0.057 | 0.538 | 0.0438 | 0.0565 | 0.0502 | 0.0352 | 0.0459 | 0.0785 | 0.2584 |

Table 4: Explanation result for both user-based and item-based K-nearest Neighbours. User keyphrases matrix was reconstructed by multiplying the user similarity or item similarity matrix

| res | R-precision | NDCG | MAP@5 | MAP@10 | MAP@50 | Precision@5 | Precision@10 | Precision@50 | Recall@5 | Recall@10 | Recall@50 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| user-based | 0.7324 | 0.8629 | 0.6919 | 0.8037 | 0.8761 | 0.6246 | 0.6789 | 0.7324 | 0.6246 | 0.6789 | 0.7324 |
| item-based | 0.2102 | 0.3134 | 0.4431 | 0.3521 | 0.2748 | 0.3561 | 0.2087 | 0.2102 | 0.3561 | 0.2087 | 0.2102 |

where $TF_{k,u}$ represent the keyphrase k's frequency in user u's review history. N is number of users in this formulation. This is done for both training and testing data. TF-IDF was used to generate the User/Item-Keyphrase table. The User-specific TF-IDF is calcualted as following:

$$TF_{k,u} = \frac{count(keyphrase - in - user)}{count(words - in - user)}$$

$$DF_k = count(keyphrase - in - document)$$

$$IDF_k = log\frac{N}{DF_k + 1}$$

$$TF - IDF = (1 + log(TF_{k,u})) * IDF_k$$

# 4 Experiments

## 4.1 Collaborative Filtering with K-nearest Neighbours

Collaborative Filtering with K-nearest Negihbours method was first tested to ensure a well-tested initial recommendation baseline. Tests were performed with both explicit and implicit user preference representation and results are generated in Table 3. Result shows empirically explicit (non-binarized) representation of user preference representation gives better recommendation results in all evaluation metrices.

## 4.2 Explanation

Explanation is the method of constructing user's keyphrase matrix by only looking at his/her user preference history (i.e. user-item matrix). We evaluate the reconstruction performance by generating the user keyphrase matrix from user preference matrix (both item-based and user-based) and compared with observed user keyphrase matrix. The result can be seen in Table 4.

## 4.3 Single Step Critiquing with user similarity latent space

In order to perform an evaluation of model's performance in a single conversational recommendation scenario using offline data provided in our datasets, we conduct an evaluation by user simulation.

Concretely, we track the performance of conversational interaction for simulated users by randomly selecting a target item from their test set, having the user critique keyphrases that appears in the target item but not the initial top recommended item, and see how the rank of target item rises. For each user, we simulated 10 target items (target items were chosen such that they appear in user test set but not training set) and for each target items, users will pick at most 10 different keyphrases with 3 different keyphrase selection methods (see following section) and evaluate their performance based on cumulative hit rate at k of the target item's rank before and after the critiquing.

### 4.3.1 Keyphrase selection method

3 Keyphrase Selection Methods were tested assuming rational users:

- **Random** - Randomly select keyphrases that are in the target item but not in the initial top recommended item.
- **Popularity** - Always choose least frequently appeared keyphrases that are in the target item but not in the initial top recommended item. Notice here, the keyphrase frequency is within all review context.
- **Difference** - Alaways select keyphrase with highest frequency difference between target item and initial top recommended item. Notice here, the keyphrase frequency are evaluated within user's review history.

### 4.3.2 Prediction method

2 different prediction methods were used for generating new recommendations based on the single step critiquing result.

- **All** - predict candidate items with regular KNN algorithm after updating latent user similarity from critiquing.
- **Upper** - Upper Bound method where candidate items without selected keyphrase are penalized to 0 prediction score.

### 4.3.3 Updating latent similarity space

Given the above methods for prediction and keyphrase selection, experiments are performed with 50 users and different $\lambda$ ratios for combining the initial latent space and critiquing space are applied:

$$modified = \lambda * initial + (1 - \lambda) * critique$$

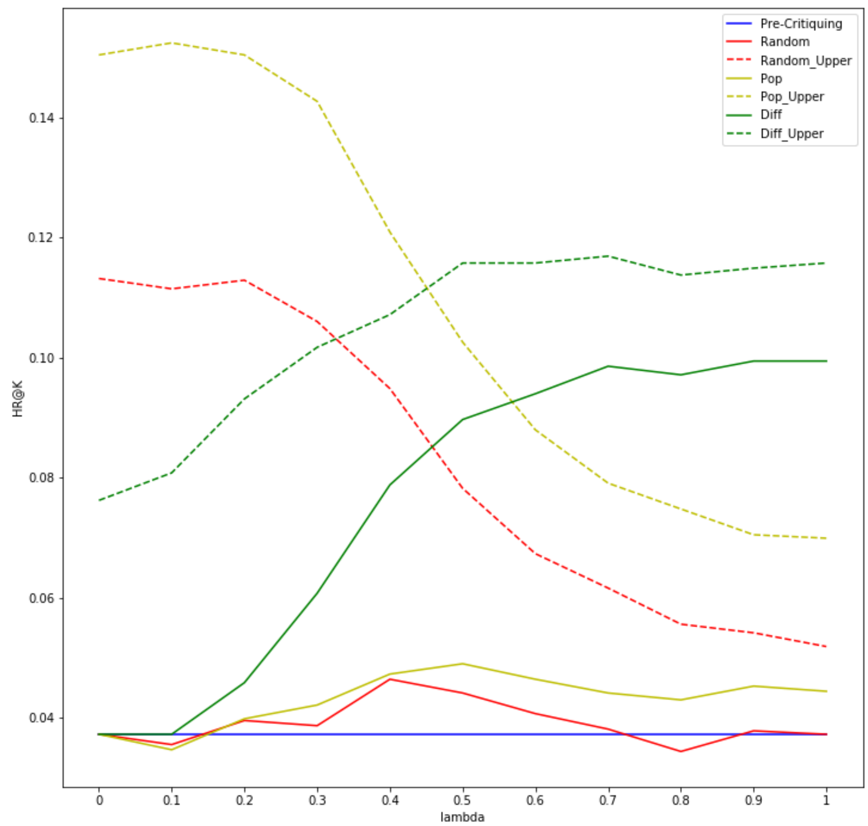


Figure 2: The HR@k vs. Lambda value performance plot given 3 different keyphrase selection method and corresponding upper bound prediction. Blue horizontal line is the hit rate for initial recommendation.

# 5 Next Steps

The next step for the thesis would focus on generalizing single-step critiquing method to multi-step critiquing where critiques given from users are not necessarily of equal importance and thus different methods for weighting/combining different steps of critiquing and the initial recommendation should be explored and tested. Moreover, the user similarity latent space could be evaluated along with another method called **Project Linear Recommendation (PLRec)**.

# Reference


1. Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: A survey and new perspectives. arXiv preprint arXiv:1804.11192 (2018).

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

3. 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.

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

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

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

7. D. Bahdanau, K. Cho and Y. Bengio. 2019. Neural Machine Translation by Jointly Learning to Align and Translate. CoRR, abs/1409.0473

8. 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.

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

10. 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.

11. 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.