EXTRA: Explanation Ranking Datasets for Explainable Recommendation

Lei Li
Hong Kong Baptist University
Hong Kong, China
csleili@comp.hkbu.edu.hk

Yongfeng Zhang
Rutgers University
New Brunswick, USA
yongfeng.zhang@rutgers.edu

Li Chen
Hong Kong Baptist University
Hong Kong, China
lichen@comp.hkbu.edu.hk

ABSTRACT

Recently, research on explainable recommender systems has drawn much attention from both academia and industry, resulting in a variety of explainable models. As a consequence, their evaluation approaches vary from model to model, which makes it quite difficult to compare the explainability of different models. To achieve a standard way of evaluating recommendation explanations, we provide three benchmark datasets for EXplanaTion RAnking (denoted as EXTRA), on which explainability can be measured by ranking-oriented metrics. Constructing such datasets, however, poses great challenges. First, user-item-explanation triplet interactions are rare in existing recommender systems, so how to find alternatives becomes a challenge. Our solution is to identify nearly identical sentences from user reviews. This idea then leads to the second challenge, i.e., how to efficiently categorize the sentences in a dataset into different groups, since it has quadratic runtime complexity to estimate the similarity between any two sentences. To mitigate this issue, we provide a more efficient method based on Locality Sensitive Hashing (LSH) that can detect near-duplicates in sub-linear time for a given query. Moreover, we make our code publicly available to allow researchers in the community to create their own datasets.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Learning to rank**.

KEYWORDS

Recommender Systems; Explainable Recommendation; Learning to Rank

ACM Reference Format:
Lei Li, Yongfeng Zhang, and Li Chen. 2021. EXTRA: Explanation Ranking Datasets for Explainable Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3404835.3463248

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463248

1 INTRODUCTION

Explainable Recommender Systems (XRS) [12, 26, 27] that not only provide users with personalized recommendations but also justify why they are recommended, have become an important research topic in recent years. Compared with other recommendation algorithms such as Collaborative Filtering (CF) [17, 18] and Collaborative Reasoning (CR) [4, 19] which aim to tackle the information overload problem for users, XRS further improves users' satisfaction and experience [20, 26, 27] by helping them to better understand the recommended items. Actually, explanation is as important as recommendation itself because usually there is no absolute "right" or "wrong" in terms of which item(s) to recommend, instead, multiple items may all be interest to the user and it all depends on how we explain our recommendation to users.

However, as explanations can take various forms, such as predefined template [9, 27], generated text [3, 10, 12] or decision paths on knowledge graphs [7, 23, 24, 28], it is sometimes difficult to evaluate the explanations produced by different methods.

In this work, with the aim to make the standard evaluation of explainable recommendation possible, we present three benchmark datasets on which recommendation explanations can be evaluated quantitatively via standard ranking metrics, such as NDCG, Precision and Recall. The idea of explanation ranking is inspired by information retrieval which intelligently ranks available contents (e.g., documents or images) for a given query. In addition, this idea is also supported by our observation on the problems of existing natural language generation techniques. In our previous work on explanation generation [10], we find that a large amount of the generated sentences are commonly seen sentences in the training data, e.g., "*the food is good*" as an explanation for a recommended restaurant. This means that the generation models are fitting the given samples rather than creating new sentences. Furthermore, even strong language models such as Transformer [21] trained on a large text corpus may generate contents that deviate from facts, e.g., "*four-horned unicorn*" [13].

Thus, we create three EXplanaTion RAnking datasets (denoted as EXTRA) for explainable recommendation research. Specifically, they are built upon user generated reviews, which are the collection of users' real feedback towards items, and thus are an ideal proxy of explanation. Also, the datasets can be further enriched by new explanations when the newly posted reviews contain new item features or up-to-date expressions.

However, simply adopting reviews [2, 6] or their sentences [5, 22] as explanations is less appropriate, because it is very unlikely for two reviews/sentences to be exactly the same, as a result, each review/sentence only appears once in the data. For this reason, almost no user shares the same review/sentence (see r1 to r6 in

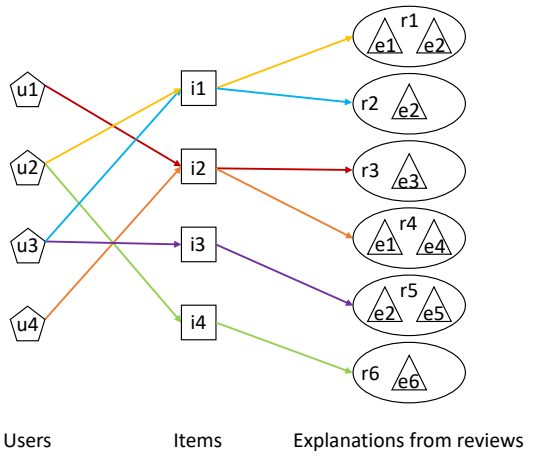


Figure 1: User-item-review interactions can be converted into user-item-explanation interactions, so as to build a connection between explanations and users/items. In the figure, r_* represents a review, and e_* represents an explanation sentence in the review.

Fig. 1), which makes it difficult to design collaborative learning algorithms for explanation ranking. Besides, not all sentences in a user review are of explanation purpose. Our solution is to extract explanation sentences from reviews that co-occur across various reviews, so as to connect different user-item pairs with one particular explanation (e.g., u2-i1 and u3-i3 with explanation e2 in Fig. 1), and build the user-item-explanation triplet interactions. By finding the commonly used sentences, the quality of explanations such as readability and expressiveness can be guaranteed. This type of textual explanations could be very effective in helping users make better and informed decisions. A recent online experiment conducted on Microsoft Office 365 [25] finds that their manually designed textual explanations, e.g., "*Jack shared this file with you*", can help users to access documents faster. It motivates us to automatically create this type of explanations for other application domains, e.g., movies, restaurants and hotels.

Then, a follow-up problem is how to detect the nearly identical sentences across the reviews in a dataset. Data clustering is infeasible to this case, because its number of centroids is pre-defined and fixed. Computing the similarity between any two sentences in a dataset is practical but less efficient, since it has a quadratic time complexity. To make this process more efficient, we develop a method that can categorize sentences into different groups, based on Locality Sensitive Hashing (LSH) [14] which is devised for near-duplicates detection. Furthermore, because some sentences are less suitable for explanation purpose (see the first review's first sentence in Fig. 2), we only keep those sentences that contain both noun(s) and adjective(s), but not personal pronouns, e.g., "*I*". In this way, we can obtain high-quality explanations that talk about item features with certain opinions but do not go through personal experiences. After the whole process, the explanation sentences remain personalized, since they resemble the case of traditional recommendation, where users of similar preferences write nearly identical review sentences, while similar items can be explained by the same explanations (see sentences in rectangles in Fig. 2).

Figure 2: Three user reviews for different movies from Amazon (Movies & TV category). Sentences of explanation purposes are highlighted in colors. Co-occurring explanations across different reviews are highlighted in rectangles.

Notice that, our datasets are different from user-item-tag data [8, 16], since a single aspect/tag when used as an explanation may not be able to clearly explain an item's specialty, e.g., a single word "*food*" cannot well describe how good a restaurant's food tastes.

To sum up, our contributions are listed below:

- We construct three large datasets consisting of user-item-explanation interactions, on which explainability can be evaluated via standard ranking metrics, e.g., NDCG. Datasets and code are made available online.[1]
- We address two key problems when creating such datasets, including the interactions between explanations and users/items, as well as the efficiency for grouping nearly identical sentences.

In the following, we first introduce our data processing approach and the resulting datasets in Section 2. Then, we present two explanation ranking formulations in Section 3. We experiment existing methods on the datasets in Section 4. Section 5 concludes this work.

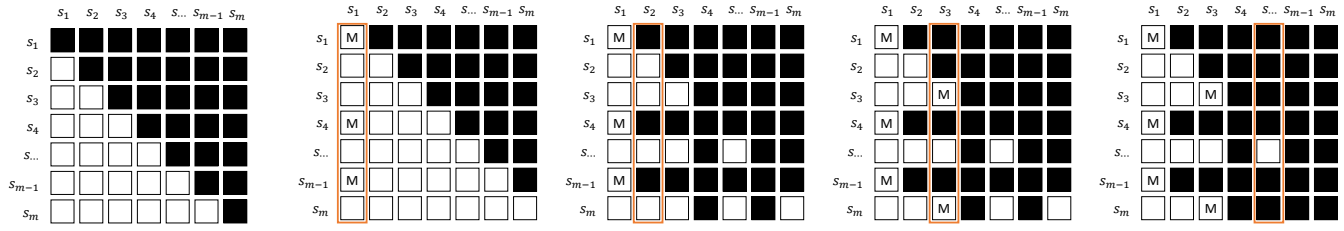
2 METHODOLOGY AND RESULTS

For explanation ranking, the datasets are expected to contain user-item-explanation interactions. In this paper, we narrow down to the explanation sentences from user reviews. The key problem is how to efficiently detect near-duplicates across different reviews, since it takes quadratic time to compute the similarity between any two sentences in a dataset. In the following, we first present our approach to finding duplicate sentences based on sentence grouping, then introduce the data construction details, and at last analyze the datasets.

2.1 Sentence Grouping

The advantage of sentence grouping is three-fold. First, it ensures the readability and expressiveness of the explanations, as they are extracted from real users' reviews based on the wisdom of the crowd. Second, it allows the explanations to be connected with both users and items, so that we can design collaborative filtering models to

[1] https://github.com/lileipisces/EXTRA



(a) Naive way (b) Sentence grouping: step 1 (c) Sentence grouping: step 2 (d) Sentence grouping: step 3 (e) Sentence grouping: step 4

Figure 3: White cells denote similarity computation, while black cells omit the computation. (a) shows a naive way to compute the similarity between any two sentences, which would take quadratic time. (b)-(e) show four example steps in our more efficient sentence grouping algorithm, where orange rectangles denote query steps in LSH, and M denotes the matched duplicates.

Algorithm 1 Sentence Grouping via LSH

Input: shingle size n, similarity threshold t, min group size g
Output: explanation set $\mathcal{E}$, groups of sentences $\mathcal{M}$

```
Pre-process textual data to obtain the sentence collection S
lsh ← MinHashLSH(t), C ← ∅
for sentence s in S do
    m ← MinHash() // create MinHash for s
    for n-shingle h in s do
        m.update(h) // convert s into m by encoding its n-shingles
    end for
    lsh.insert(m), C.add(m) // C: set of all sentences' MinHash
end for
M ← ∅, Q ← ∅ // Q: set of queried sentences
for m in C do
    if m not in Q then
        G ← lsh.query(m) // G: ID set of duplicate sentences
        if G.size > g then
            M.add(G) // only keep groups with enough sentences
            E.add(G.get()) // keep one explanation in each group
        end if
        for m' in G do
            lsh.remove(m'), Q.add(m') // for efficiency
        end for
    end if
end for
```

learn and predict such connections. Third, it makes explanation ranking and the automatic benchmark evaluation possible, since there are only a limited set of candidate explanations.

Computing the similarity between any two sentences in a dataset is computationally expensive. However, at each step of sentence grouping, it is actually unnecessary to compute the similarity for the already grouped sentences. Therefore, we can reduce the computation cost by removing those sentences (see Fig. 3 (b)-(e) for illustration). To find similar sentences more efficiently, we make use of Locality Sensitive Hashing (LSH) [14], which is able to conduct near-duplicate detection in sub-linear time. LSH consists of three major steps. First, a document (i.e., a sentence in our case) is converted into a set of n-shingles (a.k.a., n-grams). Second, the sets w.r.t. all documents are converted to short signatures via hashing, so as to reduce computation cost and meanwhile preserve document similarity. Third, the documents, whose similarity to a query document is greater than a pre-defined threshold, are returned. The detailed procedure of sentence grouping is shown in Algorithm 1.

Next, we discuss the implementation details. To make better use of all the available text in a dataset, for each record we concatenate the review text and the heading/tip. Then each piece of text is tokenized into sentences. In particular, a sentence is removed if it contains personal pronouns, e.g., "*I*" and "*me*", since explanations are expected to be more objective than subjective. We also calculate the frequency of nouns and adjectives in each sentence via NLTK[2], and only keep the sentences that contain both noun(s) and adjective(s), so as to obtain more informative explanations that evaluate certain item features. After the data pre-processing, we conduct sentence grouping via an open-source LSH [14] package Datasketch[3]. Notice that, we apply it to all sentences in a dataset, rather than that of a particular item, because it is easier to find common expressions from a large amount of sentences. When creating MinHash for each sentence, we set the shingle size n to 2 so as to preserve the word ordering and meanwhile distinguish positive sentiment from negative sentiment (e.g., "*is good*" v.s. "*not good*"). We test the similarity threshold t of querying sentences from [0.5, 0.6, ..., 0.9], and find that the results with 0.9 are the best.

2.2 Data Construction

We construct our datasets on three domains: Amazon Movies & TV[4] (movie), TripAdvisor[5] (hotel) and Yelp[6] (restaurant). In each of the datasets, a record is comprised of user ID, item ID, overall rating in the scale of 1 to 5, and textual review. After splitting reviews into sentences, we apply sentence grouping (in Algorithm 1) over them to obtain a large amount of sentence groups. A group is removed if its number of sentences is smaller than 5 so as to retain commonly seen explanations. We then assign each of the remaining groups an ID that we call an explanation ID. Eventually, a user-item pair may be connected to none, one or multiple explanation IDs since the review of the user-item pair may contain none, one or multiple explanation sentences. We remove the user-item pairs that are not

[2] https://www.nltk.org
[3] http://ekzhu.com/datasketch/lsh.html
[4] http://jmcauley.ucsd.edu/data/amazon
[5] https://www.tripadvisor.com
[6] https://www.yelp.com/dataset/challenge

Table 1: Data format of our datasets. Each dataset contains two plain text files: IDs.txt and id2exp.txt. Entries in each line of the files are separated by double colon. In IDs.txt, expID denotes the explanation ID, which corresponds to a group of near-duplicate sentences, while the senID is the original sentence ID. When a record has multiple explanation IDs, they are separated by single colon. In id2exp.txt, expID applies to both expID and senID in IDs.txt.

File	Format
IDs.txt	**userID::itemID::rating::timeStamp::expID:expID::senID:senID**
	A20YXFTS3GUGON::B00ICWO0ZY::5::1405958400::13459471:5898244::32215058:32215057
	APBZTFB6Y3TUX::B000K7VHPU::5::1394294400::13459471::21311508
id2exp.txt	**expID::expSentence**
	5898244::Great Movie
	13459471::This is a wonderful movie
	21311508::This is a wonderful movie

Table 2: Statistics of the datasets. Density is the #triplets divided by #users × #items × #explanations.

	Amazon	TripAdvisor	Yelp
# of users	109,121	123,374	895,729
# of items	47,113	200,475	164,779
# of explanations	33,767	76,293	126,696
# of (u, i) pairs	569,838	1,377,605	2,608,860
# of (u, i, e) triplets	793,481	2,618,340	3,875,118
# of explanations / (u, i) pair	1.39	1.90	1.49
Density ($\times 10^{-10}$)	45.71	13.88	2.07

connected to any explanation ID, and the remaining records are thus user-item-explanation triplets.

To make our datasets more friendly to the community, we largely follow the data format of a well-known dataset MovieLens[7]. Specifically, we store each processed dataset in two separate plain text files: **IDs.txt** and **id2exp.txt**. The former contains the meta-data information, such as user ID, item ID and explanation ID, while the latter stores the textual content of an explanation that can be retrieved via the explanation ID. The entries of each line in both files are separated by double colon, i.e., "::". If a line in IDs.txt contains multiple explanation IDs, they are separated by a single colon, i.e., ":". The detailed examples are shown in Table 1. With this type of data format, loading the data would be quite easy, but we also provide a script in our code for data loading.

2.3 Data Analysis

Table 2 shows the statistics of the processed datasets. Notice that, multiple explanations may be detected in a review, which leads to more than one user-item-explanation triplets. As we can see, all the three datasets are very sparse.

Next, we show 5 example explanations on each dataset in Table 3. We can see that the explanations vary from dataset to dataset, but they all reflect the characteristics of the corresponding datasets, e.g., "*a wonderful movie for all ages*" on the dataset Amazon Movies & TV. The occurrence of short explanations is high, not only because LSH favors short text, but also because people tend to express their opinions using common and concise phrases. Moreover, we can observe some negative expressions, which can be used to explain dis-recommendations [27].

Because constructing the datasets does not involve manual efforts, we do observe one minor issue. Since a noun is not necessarily an item feature, the datasets contain a few less meaningful explanations that are less relevant to items, e.g., "*the first time*". This issue can be effectively addressed if we pre-define a set of item features or filter out item-irrelevant nouns for each dataset.

Table 3: Example explanations after sentence grouping on three datasets. Occurrence means the number of near duplicate explanations.

Explanation	Occurrence
Amazon Movies & TV	
Excellent movie	3628
This is a great movie	2941
Don't waste your money	834
The sound is okay	11
A wonderful movie for all ages	6
TripAdvisor	
Great location	61993
The room was clean	6622
The staff were friendly and helpful	2184
Bad service	670
Comfortable hotel with good facilities	8
Yelp	
Great service	46413
Everything was delicious	5237
Prices are reasonable	2914
This place is awful	970
The place was clean and the food was good	6

3 EXPLANATION RANKING FORMULATION

The task of explanation ranking aims at finding a list of explanations to explain a recommendation for a user. Similar to item ranking, these explanations are better to be personalized to the user's interests as well as the target item's characteristics. To produce such a personalized explanation list, a recommender system can leverage the user's historical data, e.g., her past interactions

[7] https://grouplens.org/datasets/movielens/

Table 4: Performance comparison of all methods on the top-10 explanation ranking in terms of NDCG, Precision (Pre), Recall (Rec) and F1 (%). The best performing values are boldfaced.

	Amazon				TripAdvisor				Yelp			
	NDCG@10	Pre@10	Rec@10	F1@10	NDCG@10	Pre@10	Rec@10	F1@10	NDCG@10	Pre@10	Rec@10	F1@10
CD	0.001	0.001	0.007	0.002	0.001	0.001	0.003	0.001	0.000	0.000	0.003	0.001
RAND	0.004	0.004	0.027	0.006	0.002	0.002	0.011	0.004	0.001	0.001	0.007	0.002
RUCF	0.341	0.170	1.455	0.301	0.260	0.151	0.779	0.242	0.040	0.020	0.125	0.033
RICF	0.417	0.259	1.797	0.433	0.031	0.020	0.087	0.030	0.037	0.026	0.137	0.042
PITF	**2.352**	**1.824**	**14.125**	**3.149**	**1.239**	**1.111**	**5.851**	**1.788**	**0.712**	**0.635**	**4.172**	**1.068**

and comments on other items. In the following, we introduce two types of explanation ranking formulation, including global-level and item-level explanation ranking.

In the setting of **global-level explanation ranking**, there is a collection of explanations $\mathcal{E}$ that are globally shared for all items. The recommender system can estimate a score $\hat{r}_{u,i,e}$ for each explanation $e \in \mathcal{E}$ for a given pair of user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$. The user-item pair can be either an item that a user interacted before, or an item recommended for the user based on a recommendation model. According to the scores, the top-N explanations can be selected to justify why recommendation i is made for user u. Formally, this explanation list can be defined as:

$$\text{Top}(u, i, N) := \arg\max_{e \in \mathcal{E}}^{N} \hat{r}_{u,i,e} \tag{1}$$

Meanwhile, we can perform **item-level explanation ranking** to select explanations from the target item's collection, which can be formulated as:

$$\text{Top}(u, i, N) := \arg\max_{e \in \mathcal{E}_i}^{N} \hat{r}_{u,i,e} \tag{2}$$

where $\mathcal{E}_i$ is item i's explanation collection.

The two formulations respectively have their own advantages. The global-level ranking can make better use of all the user-item-explanation interactions, e.g., "*great story and acting*" for different items (see Fig. 2), so as to better capture the relation between users, items and explanations. As a comparison, item-level ranking can prevent the explanation model from presenting item-dependent explanations to irrelevant items, e.g., "*Moneyball is a great movie based on a true story*" that only applies to the movie *Moneyball*. Depending on the application scenarios, we may adopt different formulations.

4 EXPERIMENTS

In this section, we first introduce five prototype methods for explanation ranking. Then, we discuss the experimental details. At last, we analyze the results of different methods.

4.1 Explanation Ranking Methods

On the global-level explanation ranking task, we test five methods. The first one is denoted as RAND, which randomly selects explanations from the explanation set $\mathcal{E}$ for any given user-item pair. It is simply used to show the bottom line performance of explanation ranking. The other four methods can be grouped into two categories, including collaborative filtering and tensor factorization. For the ranking purpose, each of the four methods must estimate a score $\hat{r}_{u,i,e}$ for a triplet (u, i, e).

4.1.1 Collaborative Filtering. Collaborative Filtering (CF) [17, 18] is a typical type of recommendation algorithms that recommend items for a user, based on either the user's neighbors who have similar preference, or each item's neighbors. It naturally fits the explanation ranking task, as some users may care about certain item features, and some items' specialty could be similar. We extend user-based CF (UCF) and item-based CF (ICF) to our ternary data, following [8], and denote them as RUCF and RICF, where "R" means "Revised". Taking RUCF as an example, we first compute the similarity between two users u and u' via Jaccard Index as follows,

$$s_{u,u'} = \frac{|\mathcal{E}_u \cap \mathcal{E}_{u'}|}{|\mathcal{E}_u \cup \mathcal{E}_{u'}|} \tag{3}$$

where $\mathcal{E}_u$ and $\mathcal{E}_{u'}$ denote the explanations associated with u and u', respectively. Then we estimate a score for the triplet (u, i, e), for which we only retain user u's neighbors who interacted with both item i and explanation e.

$$\hat{r}_{u,i,e} = \sum_{u' \in \mathcal{N}_u \cap (\mathcal{U}_i \cap \mathcal{U}_e)} s_{u,u'} \tag{4}$$

Similarly, RICF can predict a score for the same triplet via the neighbors of items.

4.1.2 Tensor Factorization. The triplets formed by users, items and explanations correspond to entries in an interaction cube, whose missing values could be recovered by Tensor Factorization (TF) methods. Thus, we test two typical TF methods, including Canonical Decomposition (CD) [1] and Pairwise Interaction Tensor Factorization (PITF) [16]. To predict a score $\hat{r}_{u,i,e}$, CD performs element-wise multiplication on the latent factors of user u, item i and explanation e, and then sums over the resultant vector. Formally, it can be written as:

$$\hat{r}_{u,i,e} = (\mathbf{p}_u \odot \mathbf{q}_i)^\top \mathbf{o}_e = \sum_{k=1}^{d} p_{u,k} \cdot q_{i,k} \cdot o_{e,k} \tag{5}$$

where $\odot$ represents two vectors' element-wise multiplication, and d is the number of latent factors. PITF does the prediction via two sets of matrix multiplication as follows,

$$\hat{r}_{u,i,e} = \mathbf{p}_u^\top \mathbf{o}_e^U + \mathbf{q}_i^\top \mathbf{o}_e^I = \sum_{k=1}^{d} p_{u,k} \cdot o_{e,k}^U + \sum_{k=1}^{d} q_{i,k} \cdot o_{e,k}^I \tag{6}$$

where $\mathbf{o}_e^U$ and $\mathbf{o}_e^I$ are two different latent factors for the same explanation.

Table 5: Top-5 and bottom-5 explanations ranked by PITF for a user-item pair on Amazon Movies & TV dataset. There are two ground-truth explanations, and the matched one is boldfaced.

Top-5	Bottom-5
The acting is superb	Good B-movie
The cast is first rate	The final Friday
The acting is wonderful	This was a great event
The main character	Dead alive
The acting is first rate	A voice teacher and early music fan

We opt for Bayesian Personalized Ranking (BPR) criterion [15] to learn the parameters of the two TF methods, because it can model the relative ordering of explanations, e.g., the ranking score of a user's interacted explanations should be greater than that of her un-interacted explanations. The objective function of both CD and PITF is shown below:

$$\min_{\Theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u} \sum_{e \in \mathcal{E}_{u,i}} \sum_{e' \in \mathcal{E} / \mathcal{E}_{u,i}} - \ln \sigma(\hat{r}_{u,i,ee'}) + \lambda \, ||\Theta||_F^2 \qquad (7)$$

where $\hat{r}_{u,i,ee'} = \hat{r}_{u,i,e} - \hat{r}_{u,i,e'}$ denotes the difference between two interactions, $\sigma(\cdot)$ is the sigmoid function, $\mathcal{I}_u$ represents user u's interacted items, $\mathcal{E}_{u,i}$ is the explanation set of (u, i) pair for training, Θ denotes model parameters, and λ is a coefficient for preventing the model from over-fitting. To learn model parameters Θ, we optimize Eq. (7) for both CD and PITF via stochastic gradient descent. At the testing stage, we can measure scores of explanations in $\mathcal{E}$ for a user-item pair, and then rank them according to Eq. (1).

Notice that, CD and PITF may be further enriched by considering more complex relationships between explanations (e.g., the ranking score of a user's positive explanations > the other users' explanations > the user's negative explanations). We leave the exploration for future work.

4.2 Experimental Settings

As discussed earlier, this paper aims at achieving a standard way of evaluating recommendation explanations via ranking. To compare the performance of different methods on the explanation ranking task, we adopt four ranking-oriented metrics: Normalized Discounted Cumulative Gain (**NDCG**), Precision (**Pre**), Recall (**Rec**) and **F1**. Top-10 explanations are returned for each testing user-item pair. We randomly select 70% of the triplets in each dataset for training, and the rest for testing. Also, we make sure that the training set holds at least one triplet for each user, item and explanation. We do this for 5 times, and thus obtain 5 data splits, on which we report the average performance of each method.

All the methods are implemented in Python. To allow CF-based methods (i.e., RUCF and RICF) better utilize user/item neighbors, we do not restrict the upper limit of size for $\mathcal{N}_u$ and $\mathcal{N}_i$. For TF-based methods, i.e., CD and PITF, we search the number of latent factors d from [10, 20, 30, 40, 50], regularization coefficient λ from [0.001, 0.01, 0.1], learning rate γ from [0.001, 0.01, 0.1], and maximum iteration number T from [100, 500, 1000]. After parameter tuning, we use $d = 20$, $\lambda = 0.01$, $\gamma = 0.01$ and $T = 500$ for both CD and PITF.

4.3 Results and Analysis

Table 4 presents the performance comparison of different methods on three datasets. We have the following observations. First, each method performs consistently on the three datasets regarding the four metrics. Second, the performances of both RAND and CD are the worst, because RAND is non-personalized, while the data sparsity problem (see Table 2) may be difficult to mitigate for CD that simply multiplies three latent factors. Third, both RUCF and RICF that can make use of user/item neighbors, are better than RAND, but they are still limited because of the data sparsity issue. Lastly, PITF improves CD and also outperforms RUCF and RICF, with its specially designed model structure to tackle data sparsity issues (see [16] for discussions).

To better understand how explanation ranking works, in Table 5 we provide top-5 and bottom-5 explanations ranked by PITF for a specific user-item pair. For this record, there are two ground-truth explanations, i.e., "*The story is true*" and "*The cast is first rate*", and the latter is ranked the second by PITF. Moreover, the key features of the other explanations among the top-5 are "*character*" and "*acting*", which are semantically close to "*cast*". As a comparison, the bottom-5 explanations are less relevant to the ground-truth, and their sentence quality is not good. This hence shows the effectiveness of explanation ranking in finding relevant and high-quality explanations for recommendations.

5 CONCLUSION AND FUTURE WORK

In this paper, we construct three explanation ranking datasets for explainable recommendation research, with an attempt to achieve a standard way of evaluating explainability. To this end, we address two problems during data construction, including the lack of user-item-explanation interactions and the efficiency of detecting similar sentences from user reviews. Since this paper's focus is about data construction, we present our explanation ranking methods (with and without utilizing the textual content of explanations) in [11].

We believe the explanation task can be formalized as a standard ranking task just as the recommendation task. This not only enables standard evaluation of explainable recommendation but also helps to develop advanced explainable recommendation models. In the future, we will extend this work on two dimensions. One dimension is to develop multimodal explanation ranking datasets by adopting our sentence grouping approach to images, so as to construct datasets with visual explanations. Another dimension is to develop better explanation ranking models for explainable recommendation [26]. Moreover, we intend to seek industrial cooperation for conducting online experiments to test the impact of the ranked explanations to real users, e.g., the click through rate, which will help to validate explainable recommendation models under various different kinds of recommendation scenarios.

ACKNOWLEDGMENTS

This work was partially supported by Hong Kong Research Grants Council (RGC) (project RGC/HKBU12201620) and partially supported by NSF IIS-1910154 and IIS-2007907. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] J Douglas Carroll and Jih-Jie Chang. 1970. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika* 35, 3 (1970), 283–319.

[2] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *WWW*. ACM, 1583–1592.

[3] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. 2019. Generate Natural Language Explanations for Recommendation. In *SIGIR Workshop EARS*.

[4] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. *WWW* (2021).

[5] Xu Chen, Yongfeng Zhang, and Zheng Qin. 2019. Dynamic Explainable Recommendation based on Neural Attentive Models. In *AAAI*.

[6] Miao Fan, Chao Feng, Mingming Sun, and Ping Li. 2019. Reinforced product metadata selection for helpfulness assessment of customer reviews. In *EMNLP*.

[7] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, and Gerard de Melo. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *SIGIR*.

[8] Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. 2007. Tag recommendations in folksonomies. In *PKDD*. Springer.

[9] Lei Li, Li Chen, and Ruihai Dong. 2020. CAESAR: context-aware explanation based on supervised attention for service recommendations. *Journal of Intelligent Information Systems* (2020), 1–24.

[10] Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate Neural Template Explanations for Recommendation. In *CIKM*. 755–764.

[11] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Learning to Explain Recommendations. *arXiv preprint arXiv:2102.00627* (2021).

[12] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized Transformer for Explainable Recommendation. In *ACL*.

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

[14] Anand Rajaraman and Jeffrey David Ullman. 2011. Finding Similar Items. In *Mining of Massive Datasets* (3 ed.). Cambridge University Press, Chapter 3, 73–134.

[15] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[16] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*. 81–90.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW*. ACM, 175–186.

[18] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. ACM, 285–295.

[19] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. 2020. Neural Logic Reasoning. In *CIKM*.

[20] Nava Tintarev and Judith Masthoff. 2015. Explaining Recommendations: Design and Evaluation. In *Recommender Systems Handbook* (2 ed.), Bracha Shapira (Ed.). Springer, Chapter 10, 353–382.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[22] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A Reinforcement Learning Framework for Explainable Recommendation. In *ICDM*.

[23] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*. 285–294.

[24] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, and Yongfeng Zhang. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *CIKM*.

[25] Xuhai Xu, Ahmed Hassan Awadallah, Susan T. Dumais, Farheen Omar, Bogdan Popp, Robert Rounthwaite, and Farnaz Jahanbakhsh. 2020. Understanding User Behavior For Document Recommendation. In *WWW*. 3012–3018.

[26] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.

[27] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit Factor Models for Explainable Recommendation based on Phrase-level Sentiment Analysis. In *SIGIR*. ACM, 83–92.

[28] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs. In *SIGIR*.