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 (RS) 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, presents great challenges. First, user-item-explanation interactions are rare in existing RS, so how to find alternatives becomes a challenge. Our solution is to identify nearly duplicate or even 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 plan to make our code publicly available, to allow other researchers 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 ACM Conference (Conference '21), Month dd–dd, 2021, Virtual Event*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/xxxxxxx.xxxxxxx

1 INTRODUCTION

Explainable recommender systems (RS) [18, 23] that not only provide users with personalized recommendations but also justify why they are recommended, have become a popular research topic in recent years. Compared with traditional RS algorithms, e.g., collaborative filtering [16, 17], which aim to tackle the information overload problem for users, explainable RS can further improve users' satisfaction and overall experience [18] by helping them better understand recommended items. However, as explanations can take various forms, such as pre-defined template [9, 24], generated text [3, 10] and path on knowledge graph [6, 21], it is difficult to evaluate the explanations produced by different methods.

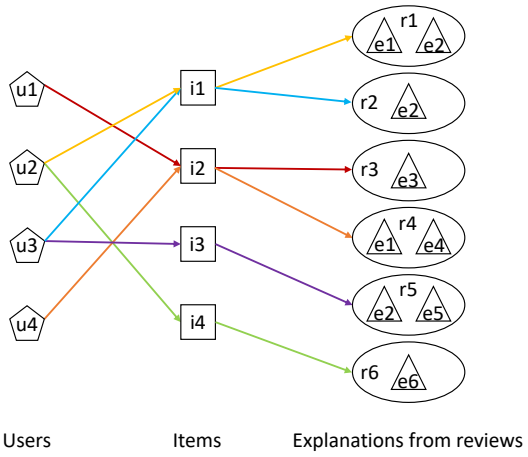


Figure 1: User-item-review interactions can be converted into user-item-explanation interactions, so as to build a connection between explanations and users/items.

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 does not create information but rather rank all the 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 natural language generation techniques. In our previous work on explanation generation [10], we find that a large amount of generated sentences are the commonly seen ones in the training data, e.g., "*the food is good*". This means that the generation models are fitting the given samples rather than creating something new. Furthermore, even strong language model as Transformer [19] trained on a large text corpus, may generate content that deviates from facts, e.g., "*four-horned unicorn*" [12].

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

Conference '21, Month dd–dd, 2021, Virtual Event

ACM ISBN 978-x-xxxx-xxxx-x/21/MM…$15.00
https://doi.org/10.1145/xxxxxxx.xxxxxxx

Figure 2: Three user reviews for different movies from Amazon (Movies & TV category). Sentences that can be regarded as explanations are highlighted in colors. Co-occurring explanations across different reviews are highlighted in rectangles.

of users' true evaluation towards items. This ensures the quality of explanations, such as readability and factuality. Moreover, the datasets could be further enriched by new explanations, when the newly added reviews contain new product features or up-to-date expressions.

However, simply adopting reviews [2, 5] or their sentences [4, 20] as explanations is less appropriate, because in this case each review/sentence only appears once, so their relation with users and items cannot be well reflected (see r1 to r6 in Fig. 1), which makes it difficult to perform explanation ranking. Our solution is to find the co-occurring sentences across all the reviews, in order to connect different user-item pairs with one particular explanation (e.g., u2-i1 and u3-i3 with e2 in Fig. 1) and thus build the user-item-explanation interactions. This type of textual explanations could be very effective in helping users make better and faster decisions. A recent online experiment conducted on Microsoft Office 365 [22] finds that their manually designed textual explanations, e.g., "*Jack share this file with you*", can help users access documents faster. It motivates us to automatically create this type of explanations for other application domains, e.g., movie.

Then, a follow-up problem is how to detect the similar or even 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 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) [13] 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 containing 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).

Notice that, our datasets are different from both user-item-aspect data [7] and user-item-tag data [8, 15], since an aspect/tag when being used as an explanation, may not be able to clearly explain an item's specialty. For example, a single word "*food*" cannot 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 codes will be made available after this paper is published.
- We address two key problems when creating such datasets, including the interactions between explanations and users/items, as well as the efficiency for grouping similar 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.

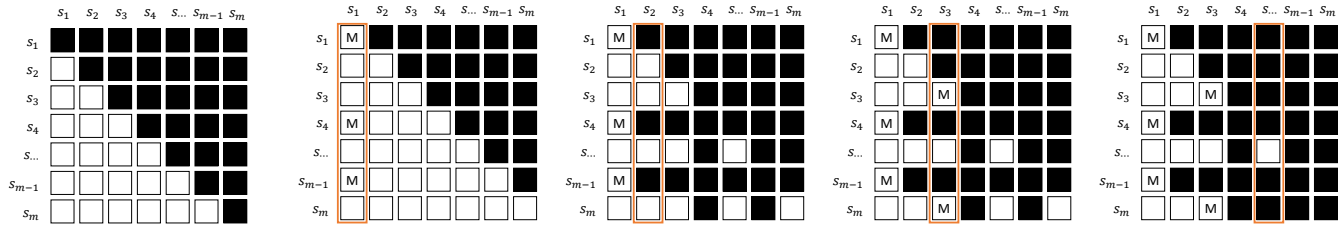
Algorithm 1 Sentence Grouping via LSH

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 $\mathcal{S}$
$lsh \leftarrow MinHashLSH(t)$, $C \leftarrow \emptyset$
for sentence s in $\mathcal{S}$ **do**
  $m \leftarrow MinHash()$ // 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
$\mathcal{M} \leftarrow \emptyset$, $\mathcal{Q} \leftarrow \emptyset$ // $\mathcal{Q}$: set of queried sentences
for m in C **do**
  if m not in $\mathcal{Q}$ **then**
    $\mathcal{G} \leftarrow lsh.query(m)$ // $\mathcal{G}$: ID set of duplicate sentences
    if $\mathcal{G}.size > g$ **then**
      $\mathcal{M}.add(\mathcal{G})$ // only keep groups with enough sentences
      $\mathcal{E}.add(\mathcal{G}.get())$ // keep one explanation in each group
    end if
    for m' in $\mathcal{G}$ **do**
      $lsh.remove(m')$, $\mathcal{Q}.add(m')$ // for efficiency
    end for
  end if
end for

2 METHODOLOGY AND RESULTS

For explanation ranking, the datasets are expected to contain user-item-explanation interactions. In this paper, we narrow down the explanations to textual 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



(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.

present our approach to find duplicate sentences that we call 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 factuality of explanations, as they are extracted from user generated reviews based on the wisdom of the crowd. Second, it allows the explanations to have connections with both users and items, so that we could design models to learn and predict such connections. Third, it makes the idea of 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, but at each step of sentence grouping, in fact it is 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 more efficiently find similar sentences, we seek to Locality Sensitive Hashing (LSH) [13] that is able to conduct near-duplicates detection in sub-linear time. LSH consists of three major steps. First, a document (i.e., a sentence in our case) is converted to 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 but preserve document similarity. Third, the documents, whose similarity to a query document is greater than a pre-defined threshold, are returned. Our detailed procedure of sentence grouping is presented 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*", because explanations are expected to be objective rather than subjective. We also calculate the frequency of nouns and adjectives in each sentence via NLTK[1], 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 [13] package Datasketch[2]. When creating MinHash for each sentence, we set the shingle size n to 2, for the purpose of relatively preserving the word order, and distinguishing 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 top of three large datasets: Amazon Movies & TV[3] (movie), TripAdvisor[4] (hotel) and Yelp[5] (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 no more than 5, in order to retain commonly seen explanations. We then assign each of the remaining groups an ID that we call explanation ID. A record may be assigned with none, one or multiple explanation IDs. We remove the records that do not have any explanation ID.

To make our datasets more friendly to the community, we largely follow the data format of a well-known dataset MovieLens[6]. 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 single colon, i.e., ":". The detailed examples are shown in Table 1. With this type of data format, loading the data is 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 means several user-item-explanation triplets. As we can see, all the three datasets are very sparse.

[1] https://www.nltk.org

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

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 after sentence grouping, while the corresponding oexpID is the original explanation ID. When a record has multiple explanation IDs, they are separated by single colon. In id2exp.txt, expID applies to both expID and oexpID in IDs.txt.

File	Format
IDs.txt	**userID::itemID::rating::timeStamp::expID:expID::oexpID:oexpID**
	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 #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, e) triplets	793,481	2,618,340	3,875,118
Density ($\times 10^{-10}$)	45.71	13.88	2.07

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 could be used to explain disrecommendations.

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. However, this would require considerable human labor, so we leave it for future work.

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 could leverage the user's historical data, e.g., her past interactions and comments on other items. In the following, we introduce two types of explanation ranking formulation, including global-level and item-level.

In the setting of **global-level explanation ranking**, there is a collection of explanations $\mathcal{E}$ that are globally shared for all items.

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

The recommender system can estimate a score $\hat{r}_{u,i,e}$ of each explanation $e \in \mathcal{E}$, for a given pair of user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$, which is resulted from either the user's behavior or a model's prediction. 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:

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

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

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

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

The two formulations respectively have their own advantages. The global-level ranking could 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 could prevent from presenting item-dependent explanations that may not be applicable to some recommendations, 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 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 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) [16, 17] 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 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 some 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) [15]. 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.

We opt for Bayesian Personalized Ranking (BPR) criterion [14] to learn the parameters of the two TF methods, because it can model the relative order of explanations, e.g., rank of a user's interacted explanations > that of her uninteracted explanations. The objective function of both CD and PITF is shown below:

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

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 relation between explanations (e.g., rank 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

To compare the performance of different methods on explanation ranking task, we adopt four 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

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**

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 that may tackle data sparsity (see [15] for discussion).

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.

In the future, we plan to adopt our sentence grouping approach to product images, so as to construct datasets with visual explanations. We will also test other ranking methods, such as those developed for tag/aspect ranking [7]. Since the focus of this work is about data construction, we present our two tensor factorization methods (with and without the textual content of explanations) for explanation ranking in [11]. Moreover, we intend to seek industrial cooperation for conducting online experiments to test the impact of the ranked explanations to users, e.g., clicking rate.

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

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

[6] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-Aware Explainable Recommendation over Knowledge Graphs. In *SIGIR*.

[7] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware Explainable Recommendation by Modeling Aspects. In *CIKM*. ACM.

[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.

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

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

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

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

[16] 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.

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

[18] 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.

[19] 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.

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

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

[22] 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.

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

[24] 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.