

Why-Not Questions & Explanations for Collaborative Filtering

Maria Stratigi, Aikaterini Tzompanaki, Kostas Stefanidis

▶ **To cite this version:**

Maria Stratigi, Aikaterini Tzompanaki, Kostas Stefanidis. Why-Not Questions & Explanations for Collaborative Filtering. WISE, Oct 2020, Amsterdam, Netherlands. 10.1007/978-3-030-62008-0_21 . hal-03172873

HAL Id: hal-03172873
https://hal.science/hal-03172873

Submitted on 18 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Why-Not Questions & Explanations for Collaborative Filtering

Maria Stratigi[1], Katerina Tzompanaki[2], and Kostas Stefanidis[1]

[1] Tampere University, Finland
{maria.stratigi,konstantinos.stefanidis}@tuni.fi
[2] ETIS Lab, UMR 8051, CY Cergy Paris University, ENSEA, CNRS F-95302, Pontoise, France
aikaterini.tzompanaki@cyu.fr

Abstract. Throughout our digital lives, we are getting recommendations for about almost everything we do, buy or consume. However, it is often the case that recommenders cannot locate the best data items to suggest. To deal with this shortcoming, they provide explanations for the reasons specific items are suggested. In this work, we focus on explanations for items that do *not* appear in the recommendations they way we expect them to, expressed in why-not questions, to aid the system engineer improve the recommender. That is, instead of offering explanations on every item proposed by the system, we allow the developer give feedback about items that were not proposed. We consider here the most traditional category of recommenders, i.e., the collaborative filtering one, and propose ways for providing explanations for why-not questions. We provide a detailed taxonomy of why-not questions on recommenders, and model-specific explanations based on the inherent parameters of the recommender. Finally, we propose an algorithm for producing explanations for the proposed why-not questions.

Keywords: explanations; why-not questions; recommendations; collaborative filtering; recommender systems

1 Introduction

Recommendations have been integrated in many of the services available to users in recent times. Although, recommenders try to accurately propose interesting items to users according to their preferences, it is often the case that they cannot locate the best data items to suggest. This can be due to many different reasons. One reason can be the cold start problem, where the system does not have enough information about a user to make accurate predictions. Another cause may be the over-specification on the part of the users. This means that a user has previously expressed a preference for a specific category and the system is unlikely to propose items that belong to a different category. Furthermore, often the systems can be misdirected due to ambiguous information on the users and

their preferences. Finally, as a system relies a lot on its hyper-parameters and thresholds, unlucky recommendations may be tight to the system's configuration.

The problem of explaining recommendations is a long-standing problem, which is most regularly approached by introducing explanations along with recommendations (e.g., [22, 16, 4]). This way, the user or the system's designer gets insights on why an item is suggested. The explanations can then vary on granularity or presentation format based on the final consumer, i.e., the final user of the recommender or the designer of the system. In this work, we expand on the concept of *post-hoc*, model-based explanations [23], i.e., explanations provided after the recommendations have been produced and based on the knowledge of the system, by exploiting the concept of why-not questions. These questions are not about why items were proposed, but why items *were not proposed*. We judge that this kind of questions are necessary for the system engineer, who needs to better understand the system and get hints on how to debug it. For example, assume a system that recommends products to users. If the engineer finds that the products of a specific company are never proposed to a user, he/she may need to understand why, and find the best way(s) to turn the situation around. This could be in the benefit of the diversity of recommendations proposed to the final user, or even for promotional campaigns of the specific company, who does not see their products proposed by the system. On the other side, asking a why-not question may not be such a straightforward task for a final user, who is totally unaware of the context or his/her preferences. However, it can still be applied in the case of a knowledgeable user, who is aware of the context of the recommendations. For instance, a female user of a career development site may wonder why she never gets suggestions for managerial positions. In this case, a why-not explanation would help the user gain trust on the system and promote its gender-fairness. In this work, we assume the system designer as the consumer of the explanations, and leave the case of the final user as a future work.

One could suggest that explaining why a certain item is not proposed is dual to explaining why all the recommended items are proposed. With the standard explanation method, a user has to go over the recommended list and understand the differences between the proposed items and the one(s) expected. This would be a time consuming, or even impossible, task, depending on the user's understanding of the data set and the recommendation model. For this reason, already existing systems that treat the 'why' aspect of explanations cannot trivially explain missing recommendations, especially without the user feedback in the form of a why-not question. By having the system answering why-not questions, this process is streamlined and not strictly dependent on the user knowledge.

In this paper, we consider the traditional paradigm of a user-based collaborative filtering recommendation system for providing explanations to why-not questions. First, we provide a detailed categorization of why-not questions characterized by three main properties: (i) the level of absenteeism that the why-not questions mention (absence or low position in the ranking of a result set), (ii) their granularity (referring to a single result or a set), and (iii) their dependency to existing recommended items. Note, that a why not question may belong to

multiple classes. Second, we provide fine-grained and personalised model-based explanations targeted for *system engineers*. The explanations are not dependent on the context of the system (e.g., social, product, PoI recommendation). We distinguish explanations between the *general* ones, based on the general setting of the problem, and the *model-specific* explanations, based on the inherent parameters of the recommendation model. Fourth, we propose an algorithmic method for computing explanations for why-not questions in collaborative filtering. Finally, we conduct a preliminary experimental study that explores the explanations space and motivates their usage by a system designer.

2 Preliminaries and Related Work

For a general setting of our recommender, assume a set of data items I and a set of users U, where each user provides ratings for a subset of I. Specifically, a user $u \in U$ rates an item $i \in I$ with a score s. The subset of users that rated an item i is denoted by U_i, whereas the subset of items rated by a user u, is denoted by I_u. For every item i not rated by a user u, the recommender estimates a relevance score, $p(u, i)$. The items with a high relevance score for u will compose the recommendation list (called also recommended items) for the user.

The literature regarding how to estimate the relevance score of an item for a user is extensive. In this work, we will focus on *collaborative filtering*, a well established recommendation approach that recommends items that users with similar preferences like (e.g., [11, 3, 13]). Specifically, the collaborative filtering (CF) approach is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. The steps of a CF algorithm to produce a list of recommendations for a user u are: (1) Find the most similar users $Peers_u$ with u by means of a similarity function $sim(u, u')$ between u and every other user u'. (2) Predict a relevance score p for each item not rated by u based on his/her similar users $Peers_u$. (3) Recommend a list R_u with the top-k items with the highest relevance score.

The first step of the CF algorithm is to compute similarities between the users. To measure the similarity $sim(u, u')$ between two users, we exploit their ratings that are available in the recommender. Several metrics appear in the related work for counting similarities between users based on ratings. We employ here the Pearson correlation measure [11], which is fast to compute and performs very well for the case of collaborative filtering. It directly calculates the correlation between two users with a score from -1 for entirely dissimilar users, to 1 for identical users. A user u' is considered similar to u if their similarity is above a threshold th and if they have rated more than $numI$ common items. We further refine the process, by keeping only the $numP$ users with the highest similarity scores. We name these users as *peers* of u, $Peers_u$. In the second step of the algorithm, we use the peers of u to predict a relevance score $p(u, i)$ for any item i that u has not yet rated. To this end, we use the *weighted sum of others ratings* [18]. We only recommend items to u if more than $numPI$ peers have rated them. In this way, we have a more robust understanding of the items' preference by the

peers. Additionally, we do not have many *false positives*, by avoiding proposing an item only liked by one (or few) peer(s), while it is unknown to the rest. In the final step, we sort all the items we have predicted a score for and return the k items with the highest score in the list R_u. Furthermore, we denote by $pos_{R_u,i}$ the index of the item i in the list R_u.

Related Work on Explanations in CF. CF explanations are typically provided based on users implicit or explicit feedback (for a survey of explanations in recommenders, refer to [20]). For example, a direct solution is to first find a set of peers for the user in question and then produce a recommendation to this user. The explanation is that the user is similar to the peers, and the peers made good ratings on the recommended item [14]. [9] compares the effectiveness of different display styles for explanations in CF. Specifically, explanations can be displayed as an aggregated histogram of the ratings of the peers, or be displayed as the detailed ratings of the peers. Alternatively, explanations can be provided by telling the user that the recommended item is similar to other items the user liked before, where several highly rated items by the user are shown as explanations [15]. To study the usefulness of explanations in recommender systems, [19] developed a prototype system to study the effect of different types of explanations. In brief, this study shows that providing appropriate explanations can benefit the recommender system over specific goals, like transparency, persuasiveness, trustworthiness and satisfaction. More recently, there exist approaches, e.g., [6, 21, 2], for generating explanations with methods using matrix or tensor factorization, where the goal is to make latent factors more tangible. From a different perspective, [8] studies the problem of computing minimum subsets of user actions to change the top-ranked recommendations in a counterfactual setup. The concept of why-not questions is used also for probabilistic range queries in [5], either by modifying the original query or by modifying the why-not set. [7] offers a similar framework for why-not questions on reverse top-k queries. To the best of our knowledge, we are the first to define and study the problem of providing explanations based on why not questions in recommender systems.

3 Why-Not Questions

In this paper, we expand on the concept of explanations in recommender systems, by exploiting the concept of why-not questions. These questions are not about why an item is recommended but why an item is not recommended in the expected way. Instead of offering explanations on every item that is proposed by the system, we allow the user to give feedback in the form of questions. For instance, in a movies recommender if the user is not satisfied with the movies list provided by the system, he/she can ask questions like: *Why were there not any comedies recommended?* The system will answer with information based on the system characteristics and the data associated with these items. This paradigm is not yet explored in recommendation systems, while it has been recently explored in other contexts like in explaining query results in relational databases [1], in reverse skyline queries [10], and briefly in machine learning systems [12].

We propose to characterise why-not questions by three main properties: (i) their level of absenteeism, (ii) their granularity, and (iii) their dependency to existing recommended items. The first property is naturally derived from the notion of false negative results, i.e., the items that should have been returned (in a certain position) but are not. The second property goes one step beyond to express groupings of missing items (that can be regarded as false negatives). The third property, corresponds to the need of the system expert to express the fact that an item that is returned (true positive) and an item that is not (false negative) should be encountered together in a result set.

First, we examine why-not questions based on *absenteeism*. In this respect, we further distinguish between (i) *total* absenteeism, and (ii) *position* absenteeism. Question such as *Why not Titanic?* belong to the *total* category, since they are about items that do not appear in the recommendation list, without a specific requirement for the position on which they should appear. Questions that ask about the ranking of items, such as *Why not rank Titanic first?* belong to the *position* category. It is evident thus, that a position absenteeism why-not question can be applied on items that *are* recommended, but still not as highly as expected. Second, we review *Granularity*. Granularity describes the level of detail of the question that is asked, distinguishing between *atomic* cases and *group* cases. In more detail, the user is able to ask questions about specific items (atomic case), such as: *Why not Titanic?*, or about set of items that share a common characteristic (group case), such as: *Why not comedies?* Third, the *Dependency* property describes items that usually appear together in the answers, or should be returned in a specific order. Example questions are *Why there are not any comedies but there are dramas?* This kind of questions fall also in the case of group recommendations, when users expect to find groups of items together. We subsequently define why-not questions in a formal way.

Definition 1. *Let I be a set of items, u a specific user of a recommender system built on I, $R_u \subseteq I$ the set of recommended items for user u by a recommender system. A why-not question is a set of the form*

$$wn = \{(m, pos, d) \mid m \in I \text{ and } pos \in \{1, \ldots, |R_u|\} \text{ and } d \in R_u\}$$

Definition 1 is general enough to cover all the cases that we mentioned before, i.e., absenteeism (m and pos), granularity, and dependency (d). Even though not explicitly apparent, a *granularity* why-not question can be derived by expanding the group to the related items in I. For example, a why-not questions of the style *Why not comedies?* can be represented by $wn = \{(Big, ,), (Zoolander, ,)\}$, given that the system can find these two comedies in its database. Moreover, if the user wants to ask one specific type of a why-not question that does not involve all three parts m, pos, and d, then he/she can leave that part empty. For example, in the case of a total absenteeism why-not question of the style *Why not Titanic?*, the corresponding wn would be $\{(Titanic, ,)\}$. In the next paragraphs we elaborate more on the different types of why-not questions that we consider and how they are expressed using Definition 1.

Next, we specify Definition 1 to express the different properties of why-not questions (absenteeism, granularity, dependency). As the *absenteeism* property is always apparent in a why-not question, we discuss both sub-categories of the *absenteeism* property (total and position), with respect to granularity and dependency. In order to keep the notation from becoming too cumbersome, we will only give the notation for set of items (the group subcategory of the granularity property). This does not affect the formalization of the why-not questions, since both granularity questions can be noted using a set format (an individual item belongs to a set that consists of just one item). For each case, we present an intuitive description, examples in the context of a movie recommendation system, and the formal expression corresponding to that case of why-not question.

– **Total Absenteeism:**
  - *Independent*: The user asks why some items do not exist in the recommendation list.
    Example-Atomic: *Why is there not Titanic?*
    Example-Group: *Why are there not any comedies?*
    Formally, an independent total absenteeism why-not question is:

$$wn_{ti} = \{(m, ,) \mid m \in I \setminus R_u\}$$

  - *Dependent*: The user asks why certain items do not exist while other (that usually appear together) exist.
    Example-Atomic: *Why is there not Titanic while there is Up?*
    Example-Group: *Why not any thrillers when there are action films?*
    Formally, a dependent total absenteeism why-not question has the form:

$$wn_{td} = \{(m, , d) \mid m \in I \setminus R_u \text{ and } d \in R_u\}$$

– **Position Absenteeism:**
  - *Independent*: The user can question the ranking of a set.
    Example-Atomic: *Why is Titanic not ranked first?*
    Example-Group: *Why are comedies not in a higher ranking?*
    Formally, an independent position absenteeism why-not question is:

$$wn_{pi} = \{(m, pos,) \mid m \in R_u \text{ and } pos_{R_u,m} < pos\}$$

  - *Dependent*: The user asks why certain items do not appear higher in the recommendation list than other recommended items.
    Example-Atomic: *Why not place Titanic before Up?*
    Example-Group: *Why not place comedies before dramas?*
    Formally, an independent position absenteeism why-not question is:

$$wn_{pd} = \{(m, pos, d) \mid m \in I \text{ and } pos > pos_{R_u,d} \text{ and } d \in R_u\}$$

4 Why-Not Explanations

To answer a why-not question, we seek to provide meaningful explanations to the system designers. By meaningful, we mean the information that is adequate to

help the designer understand why the items are not recommended in the expected way, and subsequently use this information in order to repair the system. For this reason, we split the input of the problem to distinct components that can explain - either individually or combined - the why-not question provided by the user. These components are: the input item set, the sets of all and of similar users given the user in question, the set of ranking scores, and the recommender system design (hyperparameters) itself. To accommodate the different sources of error, we define a multi-type structure, called an *explanation*, as follows:

Definition 2. *A why-not explanation for a why-not question on the recommendations of a user u is a set of parameters of the recommender system, responsible for the absence of the missing item(s) from the (specific positions of the) recommendation list.*

We distinguish between *general* explanations, which can appear in any recommender system, and *model-specific* explanations that are based on the inherent parameters of the CF recommendation model. We further describe the general and CF explanations in Sections 4.1 and 4.2, respectively, while we provide an algorithm to compute them in Section 5. We accompany the discussion with examples of why-not questions and respective possible explanations, summarized in Table 1. For clarity, we include in this table a description in natural language for the question and the explanation. The descriptions of the explanations can be regarded as the output of a statistical analysis of the resulting explanations.

4.1 General Explanations

Users, in most cases, ask about an item that does not appear in the recommendation list. So, it is very likely that this item does not exist in the database of the system. The explanation, then, is straightforward; this item is not suggested because it is unknown to the system. Another explanation emerges from the number of returned top-k items. If that number is low, then the missing item may be further down the recommendation list. However, we do not consider that the selected k may be the problem if the item is found at an index greater than $2k$, to promote other potential (model-specific) explanations. Finally, the system may produce the same score for different items. To break the ties, it adopts a specific method, e.g., it will place first the first encountered item in the database. So, when a user poses a why-not question on an item that has been neglected due to the tie-breaking method, the system may designate the tie-breaking method as a culprit. E1-E3 in Table 1 are examples of such explanations.

4.2 CF Explanations

The concept behind CF is that the system suggests items to a user that his/her similar users have liked in the past. This makes all the possible explanations revolving around the user's peers. One scenario is that none of the peers have rated an item. In this case, the item is invisible to the system and cannot be

suggested. A similar scenario is for that item to have never been rated before by any user. This again makes the item invisible to the system. Aside from these two explanations, an answer for a "Why not item A?" question is the combination of the results for the three following questions: (i) how many peers have rated it, (ii) what scores they have given it, and (iii) how similar they are to the user. If just one or two peers have rated an item, then the system ignores it, to avoid a false suggestion. If all or most of the peers have given a low score to an item, then the system, in turn, calculates a low score for it. Finally, the similarity that a peer shares with the user is also primordial. If a peer who has a high similarity with the user does not like an item, then this has an impact on that item's final predicted score. E4-E11 in Table 1 are examples of such explanations.

We represent the results of the three aforementioned questions "How many peers have rated it?", "What scores have they given it?" and "How similar are these peers to the user?" as a set of tuples of the form $(peer, score, similarity)$. Each tuple describes a peer who has rated the targeted item and consists of three values: (i) the peer's id, (ii) the score that he/she has given to the item, and (iii) the similarity shared between the peer and the user. If the set is empty then none of the peers have rated this item and we provide explanation E10 (Table 1). If the set consists of only one or two tuples (peers) then it corresponds to explanation E4. To produce the rest of the explanations, we combine into a user-friendly explanation the number of peers that have rated the item, the similarity that these peers share with the user and the scores they have given to the item.

When a user questions the item's ranking in the recommendation list, the system again checks the same information. The system answers questions like: "Why was not item A ranked higher?" by explaining the item's statistics: how many peers have rated the item, if they favored it and how similar these peers are to the user. This type of question is vague, in the sense that the user questions the general ranking of an item without comparing it to another item; the user issued an independent question. So the system treats it as if it was a total absenteeism question. For this reason, explanations E17-E19 are the same as for a total absenteeism why-not question. Another alternative for handling these types of questions is to transform them from independent to dependent by arbitrarily selecting an item in the list. We select this item according to the specifics of the question - higher or lower ranking. For example, a question like "Why was not item A ranked higher?" can be transformed into "Why not place item A before item B?". In this case, the system returns a more detailed explanation as shown in lines E20-E23 in Table 1.

Explanations to dependent why-not questions, such as "Why not item A but item B?" or "Why not place item A before item B?", are more complicated since they involve multiple items, some of which exist and others not, mixing explanations and why-not explanations. We explain the process for the first question. The second is answered in a similar way. First, we decompose the why-not question to two separate queries. The first is a part of the user's question: "Why not item A". The second query we make is "Why not item B"[3]. Intuitively,

[3] An alternative here could be to employ a solution for explaining recommendations.

WN Question	Model	WN Explanation Description	Id
Any why-not question	General/I	Item A does not exists in the database.	E1
	General/k	You asked for few items.	E2
	General/Tie	Item had the same score as another item.	E3
Why not suggest item A?	CF/numPI	Only x (<numPI) of your peers has rated this item.	E4
	CF/{$(peer, s, sim)$}	x of your peers have given a low score to this item.	E5
	CF/{$(peer, s, sim)$}	x of your most similar users have given a low score to A	E6
	CF/{$(peer, s, sim)$}	x peers like A, but y dislike it.	E7
	CF/{$(peer, s, sim)$}	All of your peers have given the item a low score.	E8
$wn = (A, ,)$	CF/numP	None of your most similar users have rated A, but x with a lower similarity have given it a high/low score.	E9
	CF/Peers	None of your peers has rated this item.	E10
	CF/S	No one has rated this item.	E11
Why not suggest item A but suggest B?	CF/numPI	x of your peers have rated item B but only y (<numPI) has rated item A.	E12
	CF/{$(peer, s, sim)$}	x of your peers like item B but dislike A.	E13
	CF/{$(peer, s, sim)$}	x peers like item B and y dislike item A.	E14
$wn = (A, , B)$	CF/numP	Your most similar peers have not rated A but have rated B.	E15
	CF/Peers	Your peers have rated B but none of them have rated A.	E16
Why is not item A ranked higher?	CF/{$(peer, s, sim)$}	x of your peers have given a low score to this item.	E17
	CF/{$(peer, s, sim)$}	x of your most similar peers have given a low score to A.	E18
$wn = \{(A, pos_{R_u A} - 1,)\}$	CF/{$(peer, s, sim)$}	x peers like A, but y dislike it.	E19
Why is not item A higher than B?	CF/{$(peer, s, sim)$}	x of your peers like item B but dislike A.	E20
	CF/{$(peer, s, sim)$}	x peers like item B and y dislike A.	E21
	CF/numP	Your most similar peers have not rated A but have rated B.	E22
$wn = \{(A, pos_{R_u B} - 1, B)\}$	CF/Peers	Your peers have rated B but none of them have rated A.	E23
Why not suggest comedies?	CF/Peers	None of your peers rated the same movie.	E24
	CF/{$(peer, s, sim)$}	Your peers dislike comedies.	E25
	CF/Peers	None of your peers has rated a comedy.	E26
	CF/{$(peer, s, sim)$}	Only x of your peers like comedies.	E27
$wn = \{(C_1, ,), ..., (C_n)\}$	CF/numP	Your most similar peers do not like comedies but x of your least similar do.	E28

Table 1: Examples of why-not questions and explanations in CF.

since the system has promoted item B to the user instead of A (either by not even suggesting A for *total* why-not questions or with a better ranking in the recommendation list for *position* why-not questions), the results of the questions "How many peers have rated it?", "What scores have they given it?" and "How similar are these peers to the user?" have higher values than the results for A. Then, we combine the answers of these two why-not questions. For example, see E12-E16 in Table 1. We choose to explain the existence of item B as a why-not explanation, because it allows us to combine the results of the two questions more effectively than if we used a generic explanation method, such as [9].

When the user formulates a *group* why-not question, for example "Why not more comedies?", the explanation that the system provides is a union of all the answers that it would have provided for individual items. For each item in the same category as the one the user asked about and a peer has preferred in the past, we formulate a why-not question. Since this can become very cumbersome for the user to consume, one can summarize the results into a user-friendly output. For instance, "Your peers like comedies, but they have not liked the same one", meaning that the peers have shown some preference for comedies, but each peer has rated a different movie. This explains why none of them were suggested. To reduce the number of questions we issue to the system, we can take into consideration the items that the peers have shown a great preference for. These items are the most important for us, since they have the highest probability to be suggested. For example, in a system where the ratings are in the range from 1 to 5, we can only consider the items that have a rating higher

than the average 2.5. We provide more explanations for a group why-not question in lines E24-E28 of Table 1.

Algorithm 1: *WNCF*

Input: item set I, user set U, user u, why-not question $\{(i,,)\}$, rating scores S, recommendation list R_u for user u, threshold $numPI$, threshold $numP$,peers of u $Peers$, relevance score function $p(u,i)$
Output: e, explanation

```
1  if i ∉ I then
2  |  e.add('I');
3  else if ∃i' : p(u,i)=p(u,i') and pos_{R_i'} ≤ k then
4  |  e.add('Tie');
5  else if i in the 2k first entries of expanded R then
6  |  e.add('k');
7  else if i has no ratings in S then
8  |  e.add('S');
9  else if at least one peer of u has rated i then
10 |  for peer ∈ Peers do
11 |  |  if peer has rated i then
12 |  |  |  e.add((peer, s(peer,i),sim(u,peer)));
13 |  if less than numPI most similar peers of u have rated i then
14 |  |  e.add('numP');
15 |  if less than numPI peers of u have rated i then
16 |  |  e.add('numPI');
17 else
18 |  for u' user in U do
19 |  |  if u' has rated i then
20 |  |  |  e.add((u', s(u',i),-));
21 |  e.add('Peers');
22 return e;
```

5 *WNCF* Algorithm

In this section, we introduce *WNCF* (standing for *Why-Not in Collaborative Filtering*), an algorithm for the computation of why-not explanations for the CF model (Algorithm 1). *WNCF* addresses total absenteeism for atomic granularity independent why-not questions. The extension of this algorithm to explain other types of why-not questions, as discussed in Section 4, is trivial for some cases, e.g., group independent why-not questions, but not for all. We postpone the extensions to future work.

As mentioned in Section 4, we first check if a general explanation can be provided; if not, we proceed with the model-specific explanations modeled in tuples representing the peers who have rated i, along with information on their rating on i and the similarity to user u. If such peers do not exist, we provide explanations based on other users who have rated i.

In more detail, *WNCF* receives as input the item set, user set, the ranking scores, and the threshold values $numPI$ and $numP$ of the CF system, as well as the why-not question for a user u. We also consider known the peers of u and the recommendation list calculated for u, as well as the relevance score function. Line 1 checks if the specific item exists in the database. If it does not, we return the explanation code I, to indicate that the source of error is the input data set. Line 3 checks if the item shares the same relevance score with another item that appears in the list. In this case, we return the explanation code Tie, to indicate that the source of error is the tie breaking method. Line 5 checks if the item appears between the kth and $2k$th entry. In this case, we return the explanation code k, to indicate the the k maybe too low. Line 7 checks if any user in the system has rated i. If none of them did, then we return the explanation code S, to indicate that there are not rating scores for i.

Lines 9-16 check the peers of the user. For every peer who has rated i, we report the score he/she has given, as well as the similarity he/she shares with u (Line 12). Then, we check the $numP$ most similar peers of the user (Line 13). If less than $numPI$ of them have rated the item, we return the code 'numP' to express that there are not enough most similar peers who have rated the item. Subsequently, we check the rest of the peers and if there were not at least $numPI$ peers who have rated i (Line 15), then we return the explanation code 'numPI'. This indicates that from all the peers of user u less than $numPI$ peers have rated this item. Finally, if none of the peers has rated this item (Line 17) we return the explanation code $Peers$ to indicate that there are no peers who have rated the item. We also return information about the users (non-peers) who have rated the item and their scores.

Overall, our rational for returning the extra information on the users (peers or not), their ratings for the item in question and their similarity to the user u for CF systems, is two-fold. First, we can compute statistics that can be easily consumed by the developers (in their raw format or as visualisations) and help them understand more about the setting. Second, we can use this information as input to a repair mechanism, which would propose changes to the system so as to make the missing item appear in the list.

6 Experiments

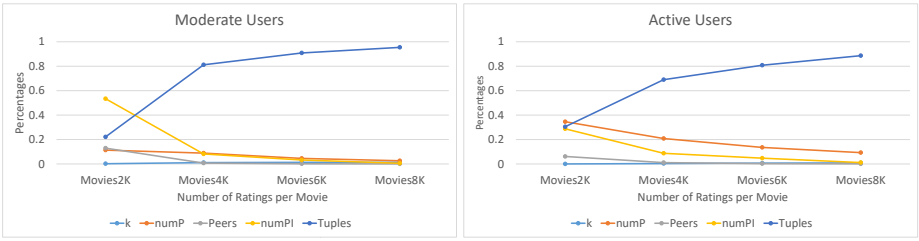
We study *WNCF* with respect to different parameters, namely the characteristics of the users for whom we pose the why-not question, and the popularity of the missing movie. Moreover, we perform an experiment that shows the next step that a developer can take, after he/she receives a *WNCF* explanation.

Experimental Setup. In the experiments we used the MovieLens 20 Million Ratings[4] that consists of 27.278 movies and 138.493 users. To study the behavior of the algorithm for different types of users we randomly selected 100 users that have rated a few items (45 to 55), called *Moderate Users*, and 100 users that have rated many items (145 to 155), called *Active Users*. To experiment with the characteristics of the movies that comprise our why-not questions we used movies of varying popularity. We randomly selected 4 sets of 100 movies that have 2K (least popular), 4K, 6K and 8K (most popular) ratings, respectively. We denote these sets as Movies2K, Movies4K, Movies6K, and Movies8K. We ensured that the movies selected are not in the recommendation lists of the users, in order to be able to run why-not questions with them. To find the peers of a user, we used the Pearson Correlation with a threshold of 0.8 (th), while to predict a score for an unrated item we utilized 100 ($numP$) peers. An item cannot be considered for addition in the recommendation least, unless 3 ($numPI$) or more peers of the user have given it a rating. We report to the user the top-10 movies with the highest predicted scores.

Explanations Study. First, we define a total absenteeism why-not question for each item in the varying popularity movie sets, for moderate and active users (Figure 1(a) and (b) respectively). Then, we run *WNCF* for each user and why-not question, and we calculate the percentage of occurrences of each explanation depending on the different parameters as they appear in the different segments of Algorithm 1. The k explanation indicates that the item in question was further down in the list that was provided to the user. The *numP* explanation means that we should augment the *numP* threshold to be able to find enough most similar peers that have rated this specific item. The *Peers* explanation occurs when none of the peers of the user has rated an item. Finally, by *Tuples* we denote explanations comprised by information on the peers of the user, calculated in Algorithm 1 line 12, and when the conditions in lines 13 and 15 are false.

Let us further analyze the result of the experiment in Figure 1. When we use the more popular movies, more of them are in the k range, almost all of them are rated by a peer of the user and most of them are rated by a top peer. This is most evident when we compare the results for Movies2K and Movies8K. For Movies2K, less than 20% of the movies could be explained by the information provided on the peers in the moderate case, while for the Movies8K more than 95% of explanations were composed by the peers. Additionally, in the Movies2K more than half of them were not rated by any of the user's peers, while in the Movies8K this number has dropped down to almost zero. We can observe similar numbers for the Active Users for the Movies2K and Movies8K movie sets. This is a self-evident result since the more popular movies have a higher chance to be rated by the peers of a user. This is further corroborated by the *Tuples* values. For the Movies2K, it has low values, since the movies are not that popular. With each subsequent movies set, as the popularity of the movies rises, so does the

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



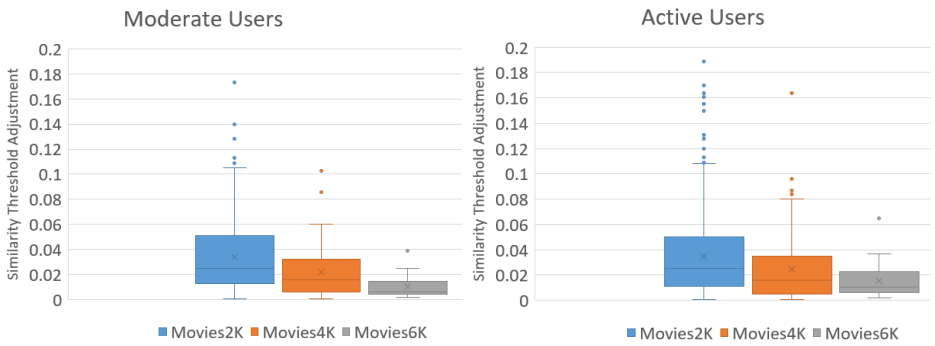
(a) (b)

Fig. 1: Explanations for varying popularity of missing movies for (a) moderate, and (b) active users.

the values of the *Tuples* variable. The most popular movies are more likely to have been voted by a top peer.

When comparing the results for the two sets of users, the Active Users have more explanations about the top peers not having rated an item (*numP*) than the Moderate Users. This is because the more ratings a user has given, the more similar other users he/she has. Since the number of peers we use is a constant variable and not a percentage, there is a higher chance the selected users have not rated the movies questioned. At the same time, we can see that the number of movies that were not rated by any of peers is lower than that of the Moderate Users. This is again because Active Users have a higher number of peers.

To demonstrate how the developer can proceed when he/she has acquired one explanation, we considered the case of *Peers* explanations. We took all the movies that were not rated by any peer (corresponding to *Peers* explanations), and we examined all the users in the system in order to find the new threshold needed in the similarity function, so as the recommender to be able to calculate a preference score for that item for the considered users. Then, we calculated the difference between the threshold we used originally (in our experiments, 0.8) and the new calculated threshold. Figures 2a and 2b show the results for the Moderate and Active users, respectively. In both experiments, we excluded the Movies8K set because the number of movies that were not rated by any of the peers is very small (less than 5 in both user sets). In both cases the adjustments needed in the similarity threshold are small. The average values (denoted with x in the figures) is below 0.04. While the Active Users have more outliers (dots in the figures) their ranges are similar to those in the Moderate user set. Finally, the median value (line inside the box) is comparable for both user sets across all movie sets. Thus, we see that with the provided explanation the developer can directly explore the right direction for debugging his system.



(a) (b)

Fig. 2: The similarity threshold (th) adjustment needed for the recommender to be able to calculate a preference score for the missing items corresponding to a *Peers* explanation for (a) moderate, and (b) active users.

7 Summary

In this work, we pay special attention on transparency provided via explanations in recommender systems. We exploit the concept of why-not questions, allowing the user to give feedback in the form of questions about why items are not proposed in the expected way. We consider the collaborative filtering approach, and propose ways for providing explanations for why-not questions. We provide a detailed taxonomy of why-not questions with respect to three main properties: (i) the level of absenteeism that the why-not questions mentions (absence or low position in the ranking of a result set), (ii) their granularity (referring to a single result or a group), and (iii) their dependency to existing recommended items. An explanation for a why-not question is meant to inform the user about the possible sources of error linked to the why-not question. We distinguish explanations between general ones, i.e., explanations that are independent to the recommendation model used, and model-specific ones, based on the inherent parameters of CF. Finally, we provide an algorithm for computing why-not explanations in CF systems. Clearly, there are many directions for future work, including proposing explanations for content-based, hybrid and sequential [17] recommendation models, as well as efficient algorithms and implementations in specific contexts. Furthermore, we target the automatic refinement of the recommendations computed for the users, by exploiting the defined explanations.

References

1. Bidoit, N., Herschel, M., Tzompanaki, K.: Immutably answering why-not questions for equivalent conjunctive queries. In: TaPP (2014)
2. Borges, R., Stefanidis, K.: On measuring popularity bias in collaborative filtering data. In: EDBT/ICDT Workshops (2020)
3. Breese, J.S., Heckerman, D., Kadie, C.M.: Empirical analysis of predictive algorithms for collaborative filtering. In: UAI (1998)
4. Chang, S., Harper, F.M., Terveen, L.G.: Crowd-based personalized natural language explanations for recommendations. In: RecSys (2016)
5. Chen, L., Gao, Y., Wang, K., Jensen, C.S., Chen, G.: Answering why-not questions on metric probabilistic range queries. In: ICDE (2016)
6. Chen, X., Qin, Z., Zhang, Y., Xu, T.: Learning to rank features for recommendation over multiple categories. In: ACM SIGIR (2016)
7. Gao, Y., Liu, Q., Chen, G., Zheng, B., Zhou, L.: Answering why-not questions on reverse top-k queries. Proc. VLDB Endow. **8**(7), 738–749 (Feb 2015)
8. Ghazimatin, A., Balalau, O.D., Roy, R.S., Weikum, G.: PRINCE: provider-side interpretability with counterfactual explanations in recommender systems. In: WSDM (2020)
9. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: CSCW (2000)
10. Islam, M.S., Zhou, R., Liu, C.: On answering why-not questions in reverse skyline queries. In: ICDE (2013)
11. Konstan, J.A., Miller, B.N., Maltz, D.A., Herlocker, J.L., Gordon, L.R., Riedl, J.: Grouplens: Applying collaborative filtering to usenet news. Commun. ACM **40**(3), 77–87 (1997)
12. Lim, B.Y., Dey, A.K., Avrahami, D.: *Why and why not* explanations improve the intelligibility of context-aware intelligent systems. In: CHI (2009)
13. Ntoutsi, E., Stefanidis, K., Rausch, K., Kriegel, H.: Strength lies in differences: Diversifying friends for recommendations through subspace clustering. In: CIKM (2014)
14. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: CSCW (1994)
15. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW (2001)
16. Stefanidis, K., Ntoutsi, E., Petropoulos, M., Nørvåg, K., Kriegel, H.: A framework for modeling, computing and presenting time-aware recommendations. Trans. Large-Scale Data- and Knowledge-Centered Systems **10**, 146–172 (2013)
17. Stratigi, M., Nummenmaa, J., Pitoura, E., Stefanidis, K.: Fair sequential group recommendations. In: SAC (2020)
18. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence (2009)
19. Tintarev, N.: Explanations of recommendations. In: RecSys (2007)
20. Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: ICDE (2007)
21. Wang, N., Wang, H., Jia, Y., Yin, Y.: Explainable recommendation via multi-task learning in opinionated text data. In: ACM SIGIR (2018)
22. Yu, C., Lakshmanan, L.V.S., Amer-Yahia, S.: Recommendation diversification using explanations. In: ICDE (2009)
23. Zhang, Y., Chen, X.: Explainable recommendation: A survey and new perspectives. Foundations and Trends in Information Retrieval **14**(1), 1–101 (2020)