On-the-spot Knowledge Refinement for an Interactive Recommender System

Yuichiro Ikemoto[1] and Kazuhiro Kuwabara[2]
[1]*Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan*
[2]*College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan*

Keywords: Knowledge Refinement, Interactive Recommender System, Crowdsourcing.

Abstract: This paper proposes a method to refine knowledge about items in an item database for an interactive recommender system. The proposed method is integrated into a recommender system and invoked when the system recognizes a problem with the item database from users' feedback about recommended items. The proposed method collects information from a user via similar interactions to those of a recommendation process. In this way, a user who is knowledgeable in a target domain, but does not necessarily know the internal system can participate in the knowledge refinement process. Thus, the proposed method paves the way for applying crowdsourcing to knowledge refinement.

1 INTRODUCTION

This paper proposes an interactive method to refine knowledge about items in the item database of a recommender system. Owning to the growing availability of large amounts of information, recommender systems are becoming increasingly popular. In this paper, we focus on a type of a recommender system that interacts with a user to obtain their preferences to provide better recommendations or to ask for a critique to improve a recommended item (e.g., (Christakopoulou et al., 2016; Widyantoro and Baizal, 2014)).

There are two main causes for recommending an unsuitable item: (1) a problem in the recommendation mechanism or (2) an error in knowledge about items in the item database. Many researches have been conducted to refine recommended items to suit a user's preferences. However, even if a recommendation algorithm, such as one predicting user's true preferences, works properly, the output of the recommendation system may be inherently incorrect if the item database contains an error.

In this paper, we address the latter issue. We let a user give feedback to the system when the user finds a wrongly recommended item. When enough feedback is accumulated, the system determines that there is a problem in the item database and invokes the *refinement mode* to collect information to identify the incorrect data and fix the error in the database.

We integrate the *refinement mode* into an interactive recommender system (Ikemoto et al., 2018). That is, the user interface in the *refinement mode* is essentially the same as that in the *recommendation mode*. The system asks the user a question about their preferences and recommends an item based on the user's acquired preferences. The user gives feedback to the system about whether the recommended item is satisfactory or inappropriate. If there are enough feedback that points out the problem, the *refinement mode* is invoked. The difference between the modes lies in how to select a question to ask and which item to present.

In the *recommendation mode*, the system asks a question that narrows down a list of possible recommended items, and in the *refinement mode*, the system asks the most promising question to help identify an error. In the *recommendation mode*, the item that most suits the user's preferences is recommended, but in the *refinement mode*, the item that may contain an error is recommended so that the system can obtain feedback from the user about the item. Since the user interaction between the user and the system is the same on the surface, we expect that a non-technical user who may not know the internal workings of the system can participate in refining the knowledge in the item database.

The remainder of this paper is organized as follows. Section 2 describes related work, and Section 3

Ikemoto, Y. and Kuwabara, K.
On-the-spot Knowledge Refinement for an Interactive Recommender System.
DOI: 10.5220/0007571508170823
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 817-823
ISBN: 978-989-758-350-6

presents the proposed method to refine an item database. Section 4 describes an example execution, and Section 5 concludes the paper.

2 RELATED WORK

To build an intelligent system, knowledge about the target domain plays a crucial role. Unless correct data are available, the system does not function properly. Several methods have been proposed to refine knowledge represented as a graph (Paulheim, 2017). Since maintaining knowledge often requires human intervention, interactive methods are effective (Atzmueller et al., 2005). Crowdsourcing is a promising approach for involving many people and has been utilized in knowledge base maintenance (Acosta et al., 2013). In particular, a gamification approach was introduced in crowdsourcing to give meaningful incentive to crowd workers (Morschheuser et al., 2017).

Gamification approaches have also been applied in linked data refinement ((Hees et al., 2011; Waitelonis et al., 2011)), and a framework to build games for this purpose has been proposed (Re Calegari et al., 2018). These works aim to bring playful elements into a tedious task.

In contrast to the aforementioned studies, we propose an approach to blend a knowledge refining task into a main task, which is an interactive recommendation. In this way, we expect increased user participation in the knowledge refinement process.

3 KNOWLEDGE REFINEMENT

3.1 Overview

The proposed knowledge refinement method is integrated into an interactive recommender system.

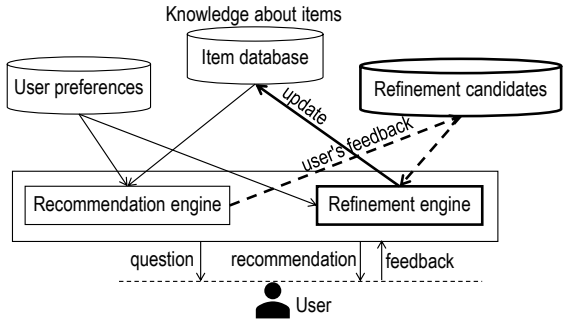

Figure 1: Overview of the proposed system.

Figure 1 shows an overview of the proposed system. The recommender system estimates a user's preferences through questions and answers, and recommends an item that suits the user's preferences. In addition to the item database and user preferences, the system has a database called *refinement candidates*, which keeps track of feedback that flags an inappropriate item recommendation.

In the following sections, we describe the data model used in the proposed system, present an underlying recommendation method, and explain the refinement method.

3.2 Data Model

There are n items in the dataset. An item is characterized by m properties. The property value of an item is either 1 (has a characteristic about of the corresponding property) or -1 (does not have a characteristics of the corresponding property).

An item s_i $(1 \leq i \leq n)$ is represented as an m-dimensional vector: $\vec{s_i} = (s_{i,1}, \cdots, s_{i,m})$, where $s_{i,j}$ represents the value of property j of item s_i and takes a value of either 1 or -1.

A user's preferences are represented as an m-dimensional vector $\vec{u} = (u_1, \cdots, u_m)$, which is initially set to $(0, 0, \cdots, 0)$. The user's response to the system's questions are recorded in the user vector.

3.3 Recommendation Mode

In the *recommendation mode*, the system asks about the user's preferences. Notably, the system asks the user if there are interested in the i^{th} property $(1 \leq i \leq m)$. If the response to the question is yes, the corresponding value of the user vector, u_i, is set to 1. For a response of no, u_i is set to -1. Otherwise, u_i remains at 0.

Each time the question is asked, the user vector is updated, and the score of items are updated, where the score represents how much an item suits the user preferences. The score of item s_i, $SCORE(s_i)$, is calculated as follows:

$$SCORE(s_i) = \sum_{j=1}^{m} u_j s_{i,j}\,. \tag{1}$$

The item with the highest score is selected, and if its score is higher than the *recommendation threshold*, it is recommended to the user. If not, another question is asked. Figure 2 shows the overall flow of the *recommendation mode*.

The order of questions is important for an efficient recommendation. As a heuristic, we calculate the information entropy of each property j $(1 \leq j \leq m)$ and

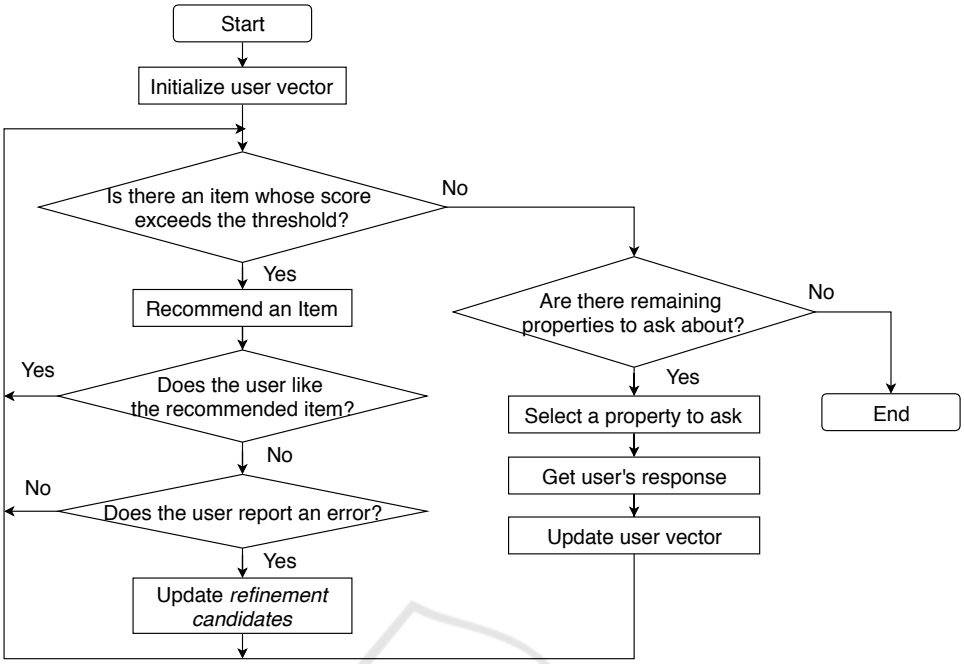

Figure 2: Recommendation mode flow.

select the property with the highest information entropy. The information entropy of property of j is calculated as follows:

$$IE_j = -\sum_{k\in\{1,-1\}} p_k \log p_k \,, \tag{2}$$

where p_k is the probability that the of value of property j is k, which is either 1 or -1. Here, we use $N_j(k)$ to denote the number of remaining items whose value of property j is k, and N to denote the total number of remaining items. If we assume that all the remaining items will be selected with equal probability, p_k can be represented as $p_k = \frac{N_j(k)}{N}$.

The reason behind this heuristic is that a property whose information entropy is higher would divide a set of items into subsets of a relatively similar size. Thus, it is expected that an item to be recommended can be identified with fewer questions.

For a recommended item, a user is expected to respond. If the user is satisfied with the recommended item, the recommendation ends. If the user asks the system to recommend another item, the system continues the search. In addition, we allow a user to give feedback that the recommended item is not appropriate. The system records that feedback in *refinement candidates*, which is almost a mirror of the item database. The value for item s_i's property j in *refinement candidates* is represented as $c_{i,j}$, which is initially set to 0, and reflects the possibility that item s_i contains an error in the value of property j.

3.4 Refinement Mode

When enough data is accumulated in *refinement candidates*, the *refinement mode* is invoked. The processing flow of the *refinement mode*, is similar to that of the *recommendation mode* except for the different selection of a question to ask and the item to be presented.

Figure 3 shows a flowchart of the *refinement mode*. First, the user vector is initialized to $(0,0,\ldots,0)$. Then, the system searches for a property to ask about. In the *recommendation mode*, the information entropy is calculated, but in the *refinement mode*, we consider *points* for each property j, $POINT(j)$, which is defined as follows:

$$POINT(j) = \sum_{i=1}^{n} c_{i,j} \,. \tag{3}$$

The value of points indicates how probable it is that an error exists in this property value of a certain item. The property with the highest points is selected; if the points equal or exceed the *point threshold*, the corresponding property is asked about. The response from the user is reflected in a user vector.

If no property can be selected for a question, we calculate an item to present and ask for a user's response. For the *refinement mode*, we define the score of item s_i as follows:

$$SCORE_r(s_i) = \sum_{j=1}^{m} u_j s_{i,j} c_{i,j} \,. \tag{4}$$

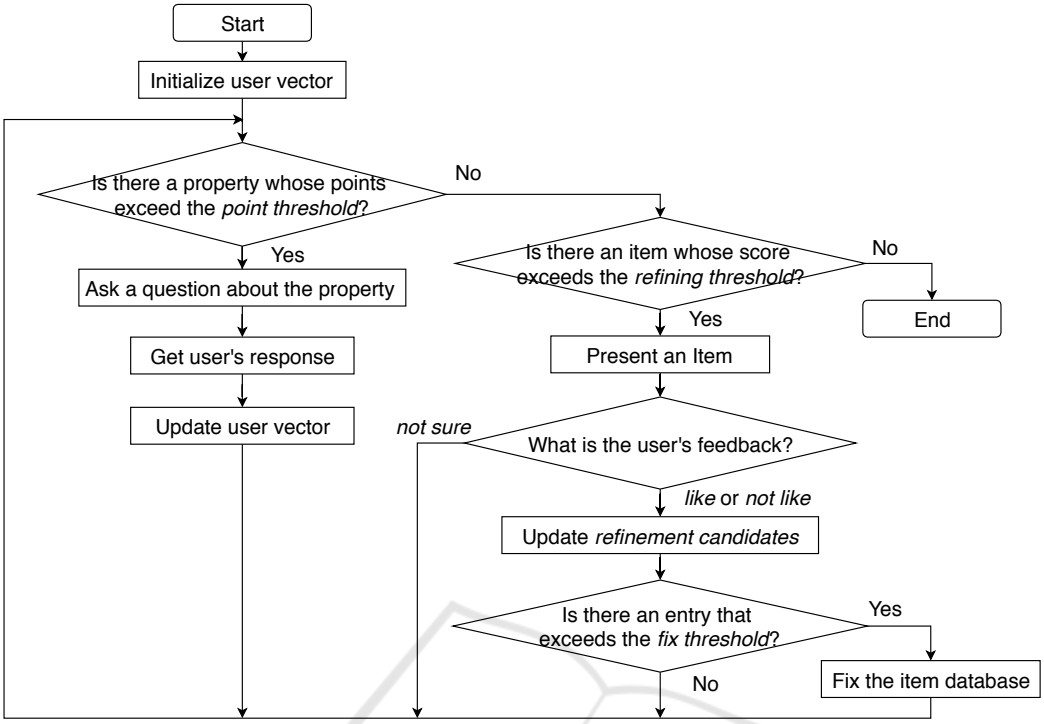

Figure 3: Refinement mode flow.

Table 1: Sample dataset (with an error in the castle property of Shimanto River).

Sightseeing spot	Property				
	nature	castle	history	summer resort	temple
Kochi Castle	-1	1	1	-1	-1
Shimanto River	1	$-1 \rightarrow 1$	-1	1	-1
Chikurin-ji Temple	-1	-1	1	-1	1
History Museum	-1	-1	1	-1	-1

Note that we consider the *refinement candidates*, so that an item that is more likely to have an error has higher precedence. The item with the highest $SCORE_r$ is selected; if the score equals or exceeds the *refining threshold*, the corresponding item is presented to a user as a recommended item and user feedback is requested, with the possible responses equating to like, not like, or not sure. Assume item s_i is presented. Then, $c_{i,j}$ $(1 \leq j \leq m)$ is updated as follows:

$$c_{i,j} \leftarrow \begin{cases} c_{i,j} - u_j s_{i,j} & \text{(if user's feedback is like)} \\ c_{i,j} + u_j s_{i,j} & \text{(if user's feedback is not like)} \\ c_{i,j} & \text{(otherwise)}. \end{cases} \tag{5}$$

Intuitively, if the user's response is like, the value of $c_{i,j}$ is decremented for the property j if the user vectors' value u_j matches the value of property j of item s_i, $s_{i,j}$; otherwise, it is incremented. If the user's response is not like, the increment and decrement are reversed.

Then, *refinement candidates* are updated. If the value of $c_{i,j}$ equals or exceeds the *fix threshold*, the corresponding value in the item database $(s_{i,j})$ is determined to be an error and is updated as follows:

$$s_{i,j} \leftarrow -s_{i,j}\ . \tag{6}$$

In this way, an error in the item database is found and fixed.

4 EXAMPLE EXECUTION

4.1 Dataset

We explain how the proposed refining method works using a simple dataset shown in Table 1. This dataset contains the data of four sightseeing spots, which were selected from popular sightseeing spots in Kochi prefecture, Japan. We define five properties to describe sightseeing spots: nature, castle, history, sum-

Table 2: Sample users' preferences.

User	Property				
	nature	castle	history	summer resort	temple
User A	1	1	-1	0	0
User B	0	1	0	1	-1
User C	1	1	0	-1	0

Table 3: Changes in scores for sightseeing spots for each user.

(a) User A

Sightseeing spot	Rounds (property asked)			
	initial	1st round (castle)	2nd round (history)	3rd round (nature)
Kochi Castle	0	1	0	-1
Shimanto River	0	1	2	3
Chikurin-ji Temple	0	-1	-2	-3
History Museum	0	-1	-2	-3

(b) User B

Sightseeing spot	Rounds (property asked)				
	initial	1st round (castle)	2nd round (history)	3rd round (summer resort)	4th round (temple)
Kochi Castle	0	1	1	0	1
Shimanto River	0	1	1	2	3
Chikurin-ji Temple	0	-1	-1	-2	-3
History Museum	0	-1	-1	-2	-1

mer resort, and temple. The property value of a sightseeing spot is set to 1 if the spot aligns with this property, and -1 otherwise. For example, the Shimanto River's property value of castle should be -1, since it has nothing to do with a castle. To show how the *refinement mode* works, we assume that the value of castle is erroneously set to 1 for Shimanto River as indicated in the grayed cell in Table 1. We will show how this error can be fixed with users participating in the process.

For the sake of explanation, let us assume that there are three users (User A, User B, and User C) as shown in Table 2. This tables shows a user's preferences, where 1 indicates that the user likes sightseeing spots with a positive corresponding, and -1 indicates that the user does not like such sightseeing spots.

4.2 Recommendation Mode

Let us assume that User A uses the system. The system starts in the *recommendation mode*. User A's user vector is initialized to $u_A = (0,0,0,0,0)$. The system calculates the scores of the sightseeing spots according to Formula (1) as shown in the initial column of Table 3(a). Since the score for all the sightseeing spots is 0, there is no sightseeing spot to be recommended. To ask the user a question, the system calculates the information entropy of each property and selects the one with the highest information entropy.

Since the number of sightseeing spots whose castle property is 1 and the number of sightseeing spots whose castle property is 0 are the same, unlike other properties, its information entropy is the highest among the properties. Thus, the system asks about the castle property. Since User A's preferences indicate that the User A likes castles, the user's response is yes, and the user vector is updated to $(0,1,0,0,0)$. Accordingly, the score of sightseeing spots is updated as in Table 3(a) (1st round column).

Let us assume that the *recommendation threshold* is 3. Since there are no sightseeing spots whose score equals or is greater than this threshold, a question about another property is asked. Since other properties have the same information entropy, a property is randomly selected. Let us assume that the next question is about history. Since User A does not like history, User A replies with no, and User A's user vector is updated to $(0,1,-1,0,0)$. Since no sightseeing spots satisfy the *recommendation threshold*, a question about another property is asked. Let us assume the next question is about nature. Since User A's preferences indicate that the user is interested in nature, the user vector is updated to $(1,1,-1,0,0)$. The scores for the sightseeing spots are updated as shown in Table 3(a) (3rd round column). In this case, the score for Shimanto River equals the *recommendation threshold* and is thus recommended to the user.

However, User A, who is assumed to like nature and castles, gives feedback that the recommended Shimanto River is inappropriate. The system then

Table 4: Refinement candidates.

(a) After *User A* used (*recommendation mode*)

Sightseeing spot	Property				
	nature	castle	history	summer resort	temple
Kochi Castle	0	0	0	0	0
Shimanto River	1	1	1	0	0
Chikurin-ji Temple	0	0	0	0	0
History Museum	0	0	0	0	0

(b) After *User B* used (*recommendation mode*)

Sightseeing spot	Property				
	nature	castle	history	summer resort	temple
Kochi Castle	0	0	0	0	0
Shimanto River	1	2	1	1	1
Chikurin-ji Temple	0	0	0	0	0
History Museum	0	0	0	0	0

(c) After *User C* used (*refinement mode*)

Sightseeing spot	Property				
	nature	castle	history	summer resort	temple
Kochi Castle	0	0	0	0	0
Shimanto River	1	3	1	1	1
Chikurin-ji Temple	0	0	0	0	0
History Museum	0	0	0	0	0

Table 5: Points of properties in the *refinement mode*.

	Property				
	nature	castle	history	summer resort	temple
Points	1	2	1	1	1

updates the *refinement candidates* as shown in Table 4(a). Since in this scenario, User A has already answered the questions about nature, castle and history with either yes or no, corresponding values are set to 1, indicating that these property values might contain an error.

Next, let us assume that User B, whose preferences are shown in Table 2, starts using the system. As with User A, User B's user vector is initialized to $(0,0,0,0,0)$, and the question about castles is asked. Since User B replies with yes, the user vector is updated to $(0,1,0,0,0)$, and the scores for sightseeing spots are updated as shown in Table 3(b) (1st round column).

Let us assume that the system asks about history next. Since User B's reply is not sure, the user vector does not change. The next question is about summer resorts. User B's reply is yes, and the user vector is updated to $(0,1,0,1,0)$. Then, the score is recalculated as shown in the Table 3(b) (2nd round column). For a question about temple, User B's reply is no, and the scores are updated as shown Table 3(b) (3rd round column). Then, the Shimanto River is recommended since the score of Shimanto River is 3. As with User A, User B's feedback is no to Shimanto River.

In this case, User B has responded to the questions castles, summer resorts, and temples with yes or no, and the corresponding values in *refinement candidates* are incremented. Note that User B's reply to the question about history is not sure, so the value regarding history does not change.

4.3 Refinement Mode

Continuing the example execution, let us assume that User C, whose preferences are shown in Table 2, starts using the system in the *refinement mode* with the *refining candidate* as shown in Table 4(b). Let us also assume that the *point threshold* is set to 2. Since the property whose points are the largest is castle, and its points satisfy this threshold (Table 5), a question is asked about this property. Since User C likes castles, they answer this question with yes. The user vector of User C is updated to $(0,1,0,0,0)$, and the $SCORE_r$ is updated as shown in Table 6.

Since the points of the other properties do not satisfy the threshold, there are no other properties to ask about, and the system looks for an item to present to the user. In this scenario, the Shimanto River has the highest score ($SCORE_r$) of 2. If we assume the *refining threshold* is 2, the Shimanto River is presented to User C as the recommended item.

Table 6: $SCORE_r$ in the *refinement mode*.

Sightseeing spot	Rounds initial
Kochi Castle	0
Shimanto River	2
Chikurin-ji Temple	0
History Museum	0

In the example dataset, the Shimanto River incorrectly has the characteristics of castle. User C, who likes castles, gives a feedback that they do not like this item. Using Formula (5), $c_{i,j}$ is updated as shown in Table 4(c). Note that, compared with Table 4(b), the castle value of Shimanto River is incremented.

Here, if we assume the *fix threshold* is 3, the castle property of Shimanto River is judged to be incorrect, and its value is set to -1, which is correct. In this way, an error in the item database can be fixed.

5 CONCLUSION

This paper proposed a method for refining an item database in a interactive recommender system. The main feature of the proposed method is that the refinement process is integrated into the recommendation process. Thus, more people can easily participate in the refinement process, and the proposed method can pave the way for using crowdsourcing for refining knowledge.

Here, we implicitly assume that users are not malicious. When we deploy the proposed method in a real-world situation, we need to deal with malicious users and user mistakes or misunderstandings, which may be a focus of future work.

We are currently building a prototype based on the proposed method. We plan to simulate a refinement process by building various user models and determine proper parameter values. Using the prototype, we will examine the effectiveness of the proposed method from the perspective of how efficiently errors in an item database can be found and repaired. We also plan to let human users interact with the system and to evaluate their subjective impression of using the system.

ACKNOWLEDGEMENTS

This work was partially supported by JSPS KAKENHI Grant Number 18K11451.

REFERENCES

Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Auer, S., and Lehmann, J. (2013). Crowdsourcing linked data quality assessment. In Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J. X., Aroyo, L., Noy, N., Welty, C., and Janowicz, K., editors, *The Semantic Web – ISWC 2013*, pages 260–276. Springer Berlin Heidelberg.

Atzmueller, M., Baumeister, J., Hemsing, A., Richter, E.-J., and Puppe, F. (2005). Subgroup mining for interactive knowledge refinement. In Miksch, S., Hunter, J., and Keravnou, E. T., editors, *Artificial Intelligence in Medicine*, pages 453–462. Springer Berlin Heidelberg.

Christakopoulou, K., Radlinski, F., and Hofmann, K. (2016). Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 815–824. ACM.

Hees, J., Roth-Berghofer, T., Biedert, R., Adrian, B., and Dengel, A. (2011). BetterRelations: Using a game to rate linked data triples. In Bach, J. and Edelkamp, S., editors, *KI 2011: Advances in Artificial Intelligence*, pages 134–138. Springer Berlin Heidelberg.

Ikemoto, Y., Asawavetvutt, V., Kuwabara, K., and Huang, H.-H. (2018). Conversation strategy of a chatbot for interactive recommendations. In Nguyen, N. T., Hoang, D. H., Hong, T.-P., Pham, H., and Trawiński, B., editors, *Intelligent Information and Database Systems ACIIDS 2018*, pages 117–126. Springer International Publishing.

Morschheuser, B., Hamari, J., Koivisto, J., and Maedche, A. (2017). Gamified crowdsourcing: Conceptualization, literature review, and future agenda. *International Journal of Human-Computer Studies*, 106(Supplement C):26–43.

Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.

Re Calegari, G., Fiano, A., and Celino, I. (2018). A framework to build games with a purpose for linked data refinement. In Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M. C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., and Simperl, E., editors, *The Semantic Web – ISWC 2018*, pages 154–169. Springer International Publishing.

Waitelonis, J., Ludwig, N., Knuth, M., and Sack, H. (2011). WhoKnows? evaluating linked data heuristics with a quiz that cleans up DBpedia. *Interactive Technology and Smart Education*, 8(4):236–248.

Widyantoro, D. H. and Baizal, Z. (2014). A framework of conversational recommender system based on user functional requirements. In *2nd International Conference on Information and Communication Technology (ICoICT 2014)*, pages 160–165. IEEE.