Improving the User Experience with a Conversational Recommender System

Fedelucio Narducci(✉), Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro

Department of Computer Science, University of Bari Aldo Moro, Via E. Orabona, 4, Bari, Italy
{fedelucio.narducci,marcode.gemmis,pasquale.lops, giovanni.semeraro}@uniba.it

Abstract. Chatbots are becoming more and more popular for several applications like customer care, health care, medical diagnoses. Generally, they have an interaction with users based on natural language, buttons, or both. In this paper we study the user interaction with a content-based recommender system implemented as a Telegram chatbot. More specifically, we investigate on one hand what are the best strategies for reducing the cost of interaction for the users and, on the other hand how to improve their experience. Our chatbot is able to provide personalized recommendations in the movie domain and implements critiquing strategies for improving the recommendation accuracy as well. In a preliminary experimental evaluation, carried out through a user study, interesting results emerged.

Keywords: Conversational recommender system · Chatbot User experience

1 Background and Motivations

The peculiarity of a conversational recommender system is its capability of interacting with the user during the recommendation process [1]. The user can provide feedback that the recommender can use for improving the next recommendation cycles. Accordingly, the acquisition of the preferences is an incremental process that might not be necessarily finalized in a single step. In fact, a cycle of interactions between the conversational recommender system and the user is repeated as long as some liked items are recommended. Hence, the goal of these systems is not only to improve the accuracy of the recommendations, but also to provide an effective user-recommender interaction.

In this paper we propose a movie recommender system implemented as Telegram chatbot[1]. Chatbots are a kind of bots which emulate user conversations.

[1] The chatbot can be tested by searching for @MovieRecSysBot in Telegram list of contacts.

We implemented a Telegram chatbot since users can interact with the system through a clean and well-known user interface daily used on their smartphone. However, our application can be easily moved to other applications (e.g. Facebook Messenger).

The entities and properties the chatbot deals with (e.g. movies, directors, actors, genres, etc.) are extracted from DBpedia[2]. The chatbot uses these properties for eliciting user preferences, for providing recommendations as well as for generating personalized explanations in natural language. The system is also capable of adapting its behavior to the user feedback by implementing a critiquing strategy proposed in [2].

The main contribution of this work is to investigate the user experience with a conversational recommender system under two points of view: the reduction of the cost of interaction for the users and the improvement of their experience. In this work the conversational recommender system adopts a user interface based on buttons.

The rest of the paper is organized as follows: the relevant literature is analyzed in Sect. 2; Sect. 3 describes how the chatbot works and its interaction with the user, and finally, the experimental evaluation and the discussion of results are reported in Sect. 4. Section 5 draws the conclusion and the future work.

2 Related Work

There is a renewed interest in conversational recommender systems in the literature. So far, the research in the field of recommender systems has been mainly focused on algorithms for improving the accuracy of rating predictions on top-n recommendations [3–5], while the presentation of recommendations to the users [6] and their interaction with the recommender have not been widely investigated.

This work is mainly focused on the analysis of strategies for improving the experience of the user with a conversational recommender. Several work analyzed the interaction between users and recommender systems under different aspects [7]. In [8], Chen and Pu argue that an easy-to-use interface is paramount in critique-based recommender systems. Berkovsky et al. [9] demonstrated that explanation and persuasion are two important characteristics for convincing users to follow the recommendations. This result is also confirmed in [10], where the highest user satisfaction is achieved by personalized explanations using item features. In [11], Kveton and Berkovsky focus their attention to devise a method that simplifies content discovery and minimizes the cost of reaching an item of interest by proposing a generalized linear search. Mahmoud and Ricci [1] demonstrate that effective conversational systems can be built by adapting the strategy for assisting online users in acquiring their goals. Similarly, in [12] the authors propose a system capable of learning how to interact with users. Christakopoulou et al. [13] develop a preference elicitation framework to identify which questions

[2] http://wiki.dbpedia.org/.

should be asked to a new user to quickly learn her preferences. In [14], the user iteratively refines a query by providing critiques like *more like item I, but smaller* for improving the recommendations.

The goal of this work, compared to the prior researches, is to study the user experience with a conversational recommender under a broader perspective that analyzes how the user interaction can be influenced by different aspects like the initial user ratings (i.e. on popular items or not), the interaction mode (i.e. by buttons or by typing), the object of preference (i.e. items or item properties), and how the users enjoy critiquing and explanation functions.

3 Description of the Chatbot

The chatbot designed in this work implements the workflow depicted in Fig. 1. In the *Preference Acquisition* step, the chatbot asks the user to express her interests. It asks questions related to entities (e.g, movies and persons) and their properties in DBpedia (e.g, genre, role, director, actors). When the user starts the interaction, her profile is empty, so the recommender system needs to address a classical cold-start problem. The system offers two different strategies to allow users express their preferences: (i) rating a set of *items* or *properties* proposed by the system; (ii) typing the entities or properties she is willing to rate. The first option allows the user to express the preferences by tapping buttons, while the second one implements an entity recognizer based on the Levenshtein distance [15] by means of a *Did you mean* function (Fig. 3(a)).

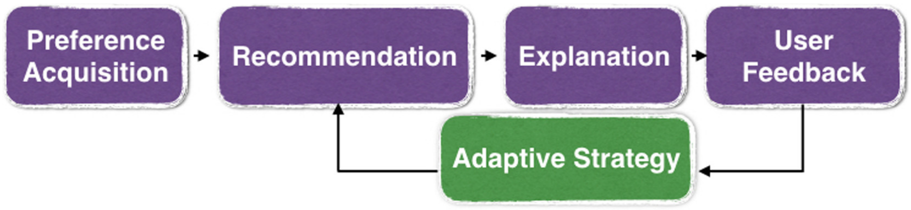


Fig. 1. The Bot workflow

The second step is the *Recommendation*. The Bot currently implements the PageRank with Priors [16], also known as Personalized PageRank. The Personalized PageRank works on a graph composed of items and properties extracted from DBpedia. In our system the nodes of the graph are entities in the movie domain like *American Beauty, Brad Pitt, Quentin Tarantino*, and the edges are the relations that connect these entities like *director, producer, actor*. The Personalized PageRank assigns different weights to different nodes to get a bias towards some nodes (in this case, the preferences of a specific user). The algorithm has been effectively used in other recommendation environments [4].

Figure 2 shows how the user preferences and the DBpedia properties are represented in a single graph. The algorithm is run for each user and the assignment of the probabilities to the nodes has been inspired by the model proposed in [17]: 80% of the total weight is evenly distributed among items and properties liked by the user (0% assigned to items disliked by the user), while 20% is evenly distributed among the remaining nodes. The algorithm generates a ranking of the items potentially interesting for a given user.

The chatbot also implements an *Explanation* component. Tintarev and Masthoff [10] point out that explaining a recommendation is generally intended as *justifying the suggestion*, but it might be also intended as *providing a detailed description* that allows the user to understand the qualities of the recommended item. The chatbot is able to provide both types of explanation. Details about an item can be obtained by tapping on the *Details* button (Fig. 3(b)) which shows information extracted from IMDB on a specific movie. The *Why?* button implements the explanation algorithm described in [18]. The idea is to use the connections in the DBpedia-based graph between the user preferences and the recommended items for explaining why a given item has been recommended.

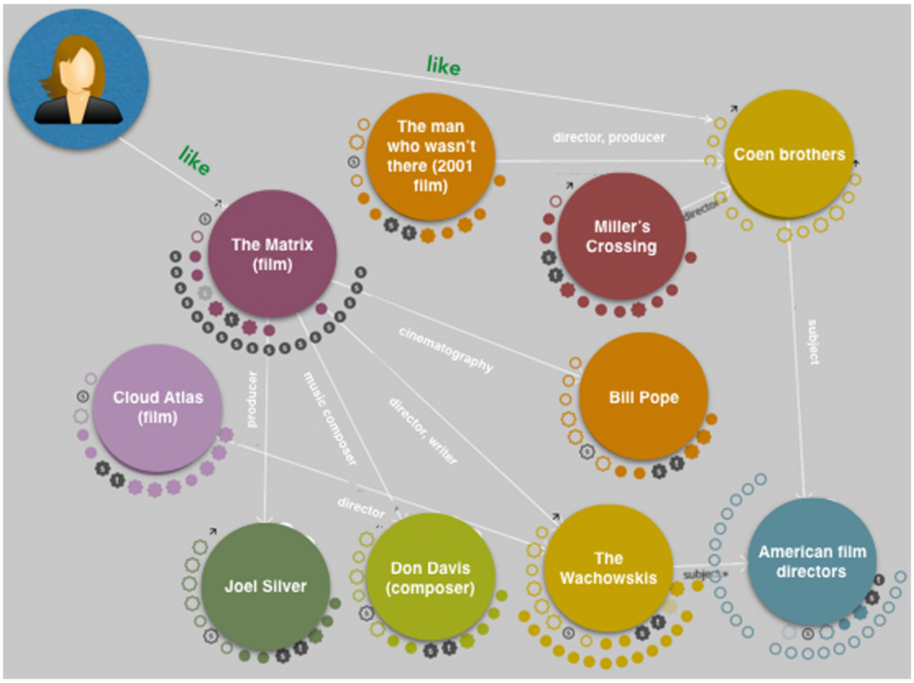
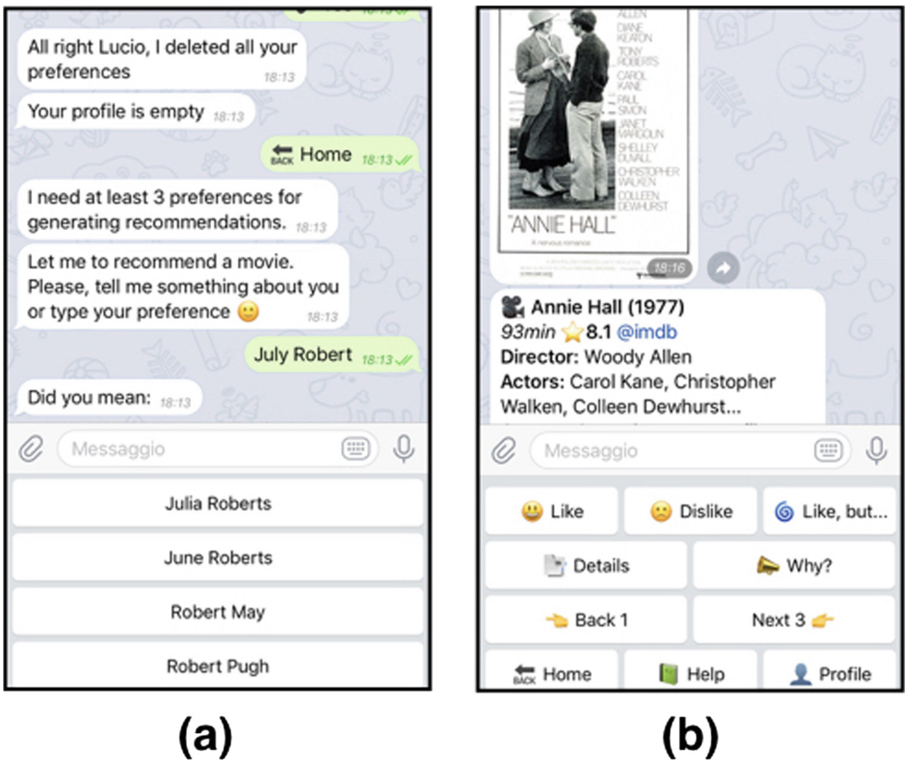


Fig. 2. Example graph which connects users, items and entities in DBpedia

An example of natural-language explanation provided by the system is: “I suggest you *Duplex* because you like movies where: the actor is *Ben Stiller* as


All right Lucio, I deleted all your preferences 18:13

Your profile is empty 18:13

Home 18:13

I need at least 3 preferences for generating recommendations. 18:13

Let me to recommend a movie. Please, tell me something about you or type your preference 18:13

July Robert 18:13

Did you mean: 18:13

Annie Hall (1977)
93min 8.1 @imdb
Director: Woody Allen
Actors: Carol Kane, Christopher Walken, Colleen Dewhurst...

Fig. 3. A screenshot of the Bot during the training phase in typing mode (a), and the recommendation phase (b)

in *Meet the Fockers*, the genre is *Comedy* as in *American Reunion*. Moreover, I recommend *Duplex* because the actor is *Ben Stiller* and you like him". In this case the system used the connections, extracted from DBpedia, between the recommended movie *Duplex* and the user preferences (i.e.*Meet the Fockers*, *American Reunion*, and *Ben Stiller*).

By tapping on the *Profile* button the user can also explore her profile, and update her preferences.

Finally, the Bot allows the user to give a feedback on a recommendation. It implements the *Adaptive Strategy* proposed in [2]. By tapping on the *Like, but...* button (Fig. 3(b)) the user activates the *Refine* process. The *Refine* is a critiquing strategy which allows the user to express a preference on a movie, but to separately evaluate its characteristics (e.g, *I like Pulp Fiction, but not Quentin Tarantino*). Therefore, the user can express a preference on a single property of a movie. The node associated to the property the user does not like (e.g., Quentin Tarantino) will be removed from the graph used by the PageRank and the recommendation process starts again on the new updated graph. The Algorithm 1 formalizes the process for leading the conversation. For the sake of simplicity, the algorithm does not report the functions for exploring and updating the profile. These are two functionalities the Bot offers to the user. Through these functions the user can view the preferences stored in her profile and change them. At the end, when the profile has been updated, the system will run again the PageRank and generate a new set of recommendations.

```
Data: Recommendations = top-5 recommendations,  Profile = set of user
      preferences,  Graph = graph representation of user preferences, items,
      entities, properties
Profile ← Profile + new preferences (items, entities, properties);
Recommendations ← PageRank (Graph, Profile);
Show Recommendations;
while User does not accept Recommendations do
    Feedback ← User feedback;
    Refine(Feedback);
    Recommendations ← PageRank (Graph, Profile);
    Show Recommendations;
end
```

Algorithm 1. Algorithm for Conversational Recommender

```
begin
    for each liked characteristics ∈ Feedback do
        Profile ← Profile + liked characteristics
    end
    for each disliked characteristics ∈ Feedback do
        remove disliked characteristic from Graph
    end
    return Graph, Profile
end
```

Procedure Refine(Feedback)

4 Experimental Evaluation

We designed a user study by involving 415 subjects (female = 36.1%, master degree or PhD = 37.9%, medium-high interest in movies = 93.2%). The subjects were recruited by sharing on Facebook, LinkedIn, and some mailing lists, the invitation to take part in the experiment. The goal of our experiment is to define the best strategies for reducing the user interaction cost and improving her experience. Our experiment has three variables:

- *the selection of the items proposed to the user in cold-start situation.* There are two different strategies: (i) the selection based on the *most popular items*, and

(ii) the selection based on the *most diverse items*[3]. Strategy (i) proposes items that the user likely knows by reducing the time for acquiring the preferences, while the strategy (ii) proposes items that are different among them (e.g. different director, or genre, or actors) in order to build a more accurate profile.
- *the interaction mode*. The user can (i) *tap* the button corresponding to the chosen answer (she must choose among the answers proposed by the system), or (ii) *type* the name of the entity or property for which she wants to express a preference (e.g. a movie, an actor, a director, a genre).
- *the preference elicitation*. There are two possibilities for eliciting preferences: (i) to express a preference on a movie (e.g. *American Beauty*), and (ii) to express a preference on a movie property (e.g. *Quentin Tarantino, Julia Roberts, comedy movies*).

By combining these variables, we obtained the four configurations reported in Table 1. Each user was randomly assigned to one of the four configurations. It is worth noting that when the user expresses her preferences on movies and properties (i.e. conf # 4) she has to type her preferences. Indeed, to make available all the possible choices by buttons was complicated. Conversely, when the user can choose only movie properties (i.e. conf # 3) the interaction through buttons is available by categorizing the properties (e.g. Actor, Director, Music Composer, etc.) and by proposing only the most popular entities for each category. This is the reason why we have four configurations instead of six.

According to the experimental protocols designed in [11,13] we adopted the following metrics for evaluating the cost of interaction: *number of questions (NQ)*, i.e. the number of questions the chatbot asked before and after the recommendation, *question-answering time (QT)*, i.e. the time (in seconds) to answer to the chatbot questions, the *interaction time (IT)* (in seconds) i.e. the time from the preference acquisition to the recommendation acceptance. In order to evaluate the accuracy of the recommender we calculated the percentage of recommended lists where at least a liked movie appears in *Liked Lists (LL)*, and the *average precision (AP@k)* computed as follows:

$$AP@k = \frac{\sum_{l=1}^{k} P@l \cdot rel(l)}{k}, \tag{1}$$

where $P@l$ is the precision considering the first l positions in the recommended list, $rel(l)$ is an indicator function equal to 1 if the item at rank l is liked, 0 otherwise. In our experiment, $k = 5$.

Experimental Protocol. We deployed a chatbot[4] designed to run a *between-subject* experiment, i.e., we tested four different configurations and each user was randomly assigned to one of them. When the user starts the experiment, her profile is empty.

[3] The diversity is computed by the Jaccard index on the movie properties between the items in the user profile and the items not rated yet.

[4] @MovieRecBot.

We asked the user to provide some basic demographic data. Then, each user follows the workflow depicted in Fig. 1. First, the user expresses her preferences according to the assigned configuration. Then the chatbot shows a set of five recommendations and for each recommended item the user can choose among *'like', 'dislike', 'like, but...', 'skip to the next movie'*. Furthermore, for each movie she can look at more details extracted from IMDB or can obtain a personalized explanation. At the end of the experiment the user gives an overall rating on her general satisfaction of the interaction with the system on a 5-point Likert scale. The recommender works on a graph composed of 42,583 nodes of which 7,672 are movies. We collected 153 users for conf #1, 131 users for conf #2, 118 users for conf #3, and 127 users for conf #4. As stated in [19], the minimum acceptable sample size for each experimental condition was set as 73, thus our experiment guaranteed the significance of the results.

Results and Discussion. Table 2 shows the results for the accuracy metrics, and Table 3 shows the results for the interaction-cost metrics. The best overall results in term of LL and AP@k is achieved by configuration #4, which is the only one where the user can type her preferences. Hence, this interaction leads to more accurate user profiles and, consequently, more accurate recommendations. Configuration #4 has also a low interaction time (IT) (i.e. 388 s). As regards the different selection mode in cold start situations, results show that when the selection is based on popularity (i.e. conf #1), a higher accuracy is achieved compared to the selection based on diverse items (i.e. conf #2). Furthermore, conf #1 has also a lower interaction time than conf #2. So, the selection based on popularity should be preferred to a selection based on diversity. The comparison between the configurations based on the preferences expressed on the items (i.e. conf # 1, 2) or on the item properties (i.e. conf # 3) shows that the latter achieves a lower accuracy as well as a larger cost of interaction. Hence, the preference elicitation based on the item properties is generally more tiring for the user and less effective. Finally, the comparison between the interactions based on buttons (i.e. conf # 1, 2, 3) and the interaction based on typing the user preferences (i.e. conf # 4) shows that the latter mode is better both in terms of accuracy and interaction cost.

By analyzing the configurations in terms of NQ and QT emerged that the number of questions is generally very similar among the different configurations. This is likely due to a quite standard procedure in the training phase. An interesting outcome emerged by analyzing the correlation between question time and interaction time: the largest time for providing answer did not imply an equally largest interaction time (i.e. conf #4), probably because the interaction mode (i.e. typing) is effective.

Other interesting statistics extracted from the analysis of the system logs shows that the critiquing strategy leads to an improvement up to +9.41% in terms of LL and up to +31.04% in terms of Ap@k. The explanation function has been used for ∼13% of recommended movies, and ∼50% of the users explored their profile during the interaction.

Table 1. The four configurations of the Bot

conf #	Rated objects	Selection	Interaction
1	Movies	Popularity	Buttons
2	Movies	Diversity	Buttons
3	Properties	–	Buttons
4	Movies/Properties	–	Typing

Table 2. Accuracy metrics - in bold the best result for each metric

conf #	LL	AP@k	Overall rating
1	0.9038	0.5662	4.14
2	0.8971	0.4958	**4.20**
3	0.8793	0.5149	3.69
4	**0.9140**	**0.5853**	4.11

Table 3. Interaction-cost metrics

conf #	NQ	QT	IT
1	20.22	**11.75**	**352**
2	21.00	14.05	390
3	21.08	14.34	412
4	**19.90**	16.76	388

Finally, the overall rating on the experience with the Bot is greater than 4 for all the configurations with the exception of conf #3 so, the users have been generally satisfied by the experience with the chatbot.

5 Conclusion and Future Work

In this paper we proposed a movie recommender system implemented as Telegram chatbot. We evaluated different interaction modes in terms of interaction cost and recommendation accuracy. We implemented a critiquing strategy which adapts the recommendations to the user feedback. Results obtained by carrying out a user study demonstrated that when the user can type her preferences, the recommender shows the best trade off between accuracy and cost of interaction. An interaction based on buttons should propose popular items for reducing the interaction cost and improving the recommendation accuracy. Furthermore, the preferences given on the items are generally more effective than the preferences given on the item properties. Other interesting outcomes are that the critiquing strategies can lead to very significant improvements in terms of recommendation accuracy, and that the explanation and the profile-exploration functions are features liked by the users.

As future work, we will investigate an interaction completely based on natural language and we will test the recommender on other domains like music and book.

Acknowledgment. This work has been funded by the projects UNIFIED WEALTH MANAGEMENT PLATFORM - OBJECTWAY SpA - Via Giovanni Da Procida nr. 24, 20149 MILANO - c.f., P. IVA 07114250967, and PON01 00850 ASK-Health (Advanced system for the interpretations and sharing of knowledge in health care).

References

1. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, pp. 73–82. ACM (2009)
2. Mcginty, L., Smyth, B.: Adaptive selection: an analysis of critiquing and preference-based feedback in conversational recommender systems. Int. J. Electron. Commer. **11**(2), 35–57 (2006)
3. Lops, P., De Gemmis, M., Semeraro, G., Narducci, F., Musto, C.: Leveraging the LinkedIn social network data for extracting content-based user profiles. In: RecSys 2011 - Proceedings of the 5th ACM Conference on Recommender Systems, pp. 293–296 (2011)
4. Basile, P., Musto, C., de Gemmis, M., Lops, P., Narducci, F., Semeraro, G.: Content-based recommender systems + DBpedia knowledge = semantics-aware recommender systems. In: Presutti, V., et al. (eds.) SemWebEval 2014. CCIS, vol. 475, pp. 163–169. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12024-9_21
5. Musto, C., Narducci, F., Lops, P., de Gemmis, M.: Combining collaborative and content-based techniques for tag recommendation. In: Buccafurri, F., Semeraro, G. (eds.) EC-Web 2010. LNBIP, vol. 61, pp. 13–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15208-5_2
6. Felfernig, A., Burke, R., Pu, P.: Preface to the special issue on user interfaces for recommender systems. User Model. User-Adapt. Interact. **22**(4), 313–316 (2012)
7. Narducci, F., Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Leveraging encyclopedic knowledge for transparent and serendipitous user profiles. In: Carberry, S., Weibelzahl, S., Micarelli, A., Semeraro, G. (eds.) UMAP 2013. LNCS, vol. 7899, pp. 350–352. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38844-6_36
8. Chen, L., Pu, P.: Critiquing-based recommenders: survey and emerging trends. User Model. User-Adapt. Interact. **22**(1–2), 125–150 (2012)
9. Berkovsky, S., Freyne, J., Oinas-Kukkonen, H.: Influencing individually: fusing personalization and persuasion. ACM Trans. Interact. Intell. Syst. (TiiS) **2**(2), 9 (2012)
10. Tintarev, N., Masthoff, J.: Evaluating the effectiveness of explanations for recommender systems. User Model. User-Adapt. Interact. **22**(4–5), 399–439 (2012)
11. Kveton, B., Berkovsky, S.: Minimal interaction content discovery in recommender systems. ACM Trans. Interact. Intell. Syst. (TiiS) **6**(2), 15 (2016)
12. Sun, Y., Zhang, Y., Chen, Y., Jin, R.: Conversational recommendation system with unsupervised learning. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 397–398. ACM (2016)

13. Christakopoulou, K., Radlinski, F., Hofmann, K.: Towards conversational recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 815–824. ACM (2016)
14. Smyth, B., McGinty, L., Reilly, J., McCarthy, K.: Compound critiques for conversational recommender systems. In: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2004, pp. 145–151. IEEE Computer Society, Washington, DC (2004). https://doi.org/10.1109/WI.2004.45
15. Yujian, L., Bo, L.: A normalized levenshtein distance metric. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1091–1095 (2007)
16. Haveliwala, T.H.: Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search. IEEE Trans. Knowl. Data Eng. **15**(4), 784–796 (2003)
17. Musto, C., Lops, P., Basile, P., de Gemmis, M., Semeraro, G.: Semantics-aware graph-based recommender systems exploiting linked open data. In: Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, pp. 229–237. ACM (2016)
18. Musto, C., Narducci, F., Lops, P., De Gemmis, M., Semeraro, G.: ExpLOD: a framework for explaining recommendations based on the linked open data cloud. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 151–154. ACM (2016)
19. Knijnenburg, B.P., Willemsen, M.C.: Evaluating recommender systems with user experiments. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 309–352. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_9