An explanation-based approach for experiment reproducibility in recommender systems

Nikolaos Polatidis[1], Antonios Papaleonidas[2], Elias Pimenidis[3] and Lazaros Iliadis[2]

[1]School of Computing, Engineering and Mathematics, University of Brighton, BN2 4GJ, Brighton, United Kingdom
N.Polatidis@Brighton.ac.uk

[2]Department of Civil Engineering, School of Engineering, Democritus University of Thrace, Xanthi 67100, Greece
Antonis.pap@gmail.com
liliadis@civil.duth.gr

[3]Department of Computer Science and Creative Technologies, University of the West of England, BS16 1QY, Bristol, United Kingdom
Elias.Pimenidis@uwe.ac.uk

Abstract

The offline evaluation of recommender systems is typically based on accuracy metrics such as the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) for error rating prediction and Precision and Recall for measuring the quality of the top-N recommendations. However, it is difficult to reproduce the results since there are various libraries that can be used for running experiments and also within the same library there are many different settings that if not taken into consideration when replicating the results might vary. In this paper, we show that within the use of the same library an explanation-based approach can be used to assist in the reproducibility of experiments. Our proposed approach has been experimentally evaluated using a wide range of recommendation algorithms ranging from collaborative filtering to complicated fuzzy recommendation approaches that can solve the filter bubble problem, a real dataset and the results show that it is both practical and effective.

Keywords: Recommender Systems; Explanations; Reproducibility; Fuzzy logic

1. Introduction

Recommender systems are widely known for their use in e-Commerce for recommending products to users, thus reducing the overall searching time of the user and increase sales. Furthermore, it is a technology used in various other less known domains such as music recommendation or people to people recommendation in social media [1,2]. However, the increasing use and popularity of recommender systems research both in academia and in industry has led to the development of new algorithms and their experimental evaluation. Reproducing results is desirable but the complexity of doing so gives rise to potential problems that are considered as quite important [3]. For the offline evaluation of recommender systems various metrics can be used such as MAE and RMSE for predicting the accuracy error and information retrieval metrics such as Precision and Recall can be used for measuring the quality of the top-N recommendations [4]. While there are more metrics, it is outside the scope of this paper to discuss them while further details can be found in [4]. In the literature there are various libraries that can be used for developing and testing a recommendation algorithm and include

Recommender101, Apache Mahout, LensKit and MyMediaLite among others [2] [5]. In the work by [2] it has been shown that reproducing the experimental results of an algorithm is very difficult when using a different library because of different settings and parameters that exist between them. However, it is shown that if a set of carefully selected guidelines is followed with the use of the same library then the results can be replicated with a very small and non-noticeable difference in the output value.

In our previous conference version, we have shown that an explanation-based approach can be used to assist in solving the problem of experiment reproducibility in recommender systems when the the same recommendation library and simple algorithms are used [1]. In this article we extend our previous paper by providing:

- A complicated fuzzy-based recommendation algorithm that can be used as a pre-processing step for different recommendation methods.
- Evaluated the above algorithm using the previously developed explanation-based approach, a publicly available recommendation library, different recommendation methods and a real dataset, with the results indicating that the use of explanation can reduce the evaluation cycles necessary to achieve reproducible results in complicated scenarios.

The rest of the paper is organized as follows: Section 2 provides the relevant background, section 3 delivers the proposed approach, section 4 presents the experiments and section 5 contains the discussion and section 6 is the conclusions and future work part.

2. Related work

Evaluating recommender systems in offline environments can be done using prediction accuracy or information retrieval metrics. However, the problem arises when in a research output of a new algorithm the source code is not made publicly available or when the exact settings for replicating the code and the experiments are missing. In the literature there are related works that have done important steps towards the solution of the reproducibility problem. In [2] a very good analysis of the main problems is identified, which include the name and the source code of the recommendation library, the details of the algorithm, the dataset used and the details of how the dataset has been used. Moreover, in the same work a set of guidelines is proposed that can be followed to assist in the reproducibility. Another similar work that identifies a set of best practices for recommender systems can be found in [6], while in [7] the importance of the reproducibility of experiments in recommender systems evaluation was highlighted with the organization of a workshop in 2013. Furthermore, the outcome of this workshop can be found in a relevant report with its future directions being theoretical only [8]. One other relevant approach can be found in [9] and it is about the improvement of the statistical power of the 10-fold cross validation scheme in recommender systems. A more relevant but more software-oriented approach is Rival [10]. In this approach a toolkit provided different stages in the process such as data splitting, item recommendation and evaluation. It is not a framework or a library but a toolkit that can be used in Apache Mahout, LensKit and MyMediaLite and it provides a user interface. Other researchers having known about the reproducibility problem have decided to develop and propose their own evaluation metrics. For example, in [11] the authors proposed a general evaluation metric that operates over a set of sessions, while another proposed metric can be found in [12] where the authors propose the modified Reciprocal Hit Rand Metric (mRHR) which is a hit rank metric.

In addition to the related works, the most common recommendation method is Collaborative Filtering (CF) and the most known CF method is Pearson Correlation Coefficient (PCC). PCC is defined in equation 1 and Sim (a, b) is the similarity between users a and b, also $r_{a,p}$ is the rating of user a for product p, $r_{b,p}$ is the rating of user b for product p and $\acute{r}a$ and $\acute{r}b$ represent the user's average ratings. P is the set of all products. Moreover, the similarity value ranges from -1 to 1 and higher is better.

$$PCC_{a,b} = \frac{\sum p \in P(ra, p - \acute{r}a)(rb, p - \acute{r}b)}{\sqrt{\sum p \in P(ra, p - \acute{r}a)^2}\sqrt{\sum p \in P(rb, p - \acute{r}b)^2}} \quad (1)$$

Furthermore, to measure the prediction error, MAE it is typically be used and is defined in equation 2 where *pi* is the predicted rating and *ri* is the actual rating in the summation. This method is used for the computation of the deviation between the predicted ratings and the actual ratings. It should also be noted that lower values are better.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |pi - ri| \quad (2)$$

However, there are numerous settings found in a recommendation library that can affect the result, such as the number of the nearest neighbors, if a cross-fold evaluation took place or the dataset what split into a training and testing part and the minimum ratings per item or if a threshold of minimum ratings that a user has submitted for an item will be applied. In table 1 we can see the results of PCC using different neighborhood size, the MovieLens 1 million dataset [13], 80% training and 20% testing with the use of the Recommender101 library. Furthermore, in table 2 it is shown that if the minimum number of ratings per user is different the output can vary significantly on 5-fold cross validation. For table 1 the cross validation took place 3 times and the average result is presented.

	Number of k nearest neighbours					
	60	80	100	200	300	400
PCC	0.870	0.862	0.841	0.811	0.785	0.761

Table 1. MAE results for Recommender101

Settings	Min number of ratings per user (30)	Min number of ratings per user (Not known and not specified – Default value used by the library)
PCC	0.872	0.890

Table 2. 5-fold cross-fold with different settings MAE results based on Recommender101

3. Proposed approach

Previously it has been shown that it is very difficult to reproduce results using different evaluation libraries due to the differences that exist between the implementations of algorithms and metrics [2]. However, with the use of the same library the possibility of correctly reproducing an algorithm and an experimental evaluation is high if the same settings and parameters are used.

For our proposed approach we use the Recommender101 library in combination with a set of explanations that accompany the output log file of the result. The library comprises a set of components

for offline evaluation as shown in figure 1. The settings used such as, the algorithm applied, the number nearest neighbors, type of validation (cross fold or test/train) and the algorithm evaluated are passed in an external configuration file. Moreover, it supports well known metrics such as MAE, RMSE, Precision, Recall, NDCG among others and when the experiment is finished the result is printed on the screen and saved in a log file. We extend the Recommender101 library to print on the screen and also save in the log file a set of explanations in simple language that can be used to guide a future researcher to reproduce an experiment.

In addition to the settings used it should be noted that it is difficult to exactly replicate an experiment since in most cases there are many settings and parameters that if not mentioned it is challenging to reproduce a result.

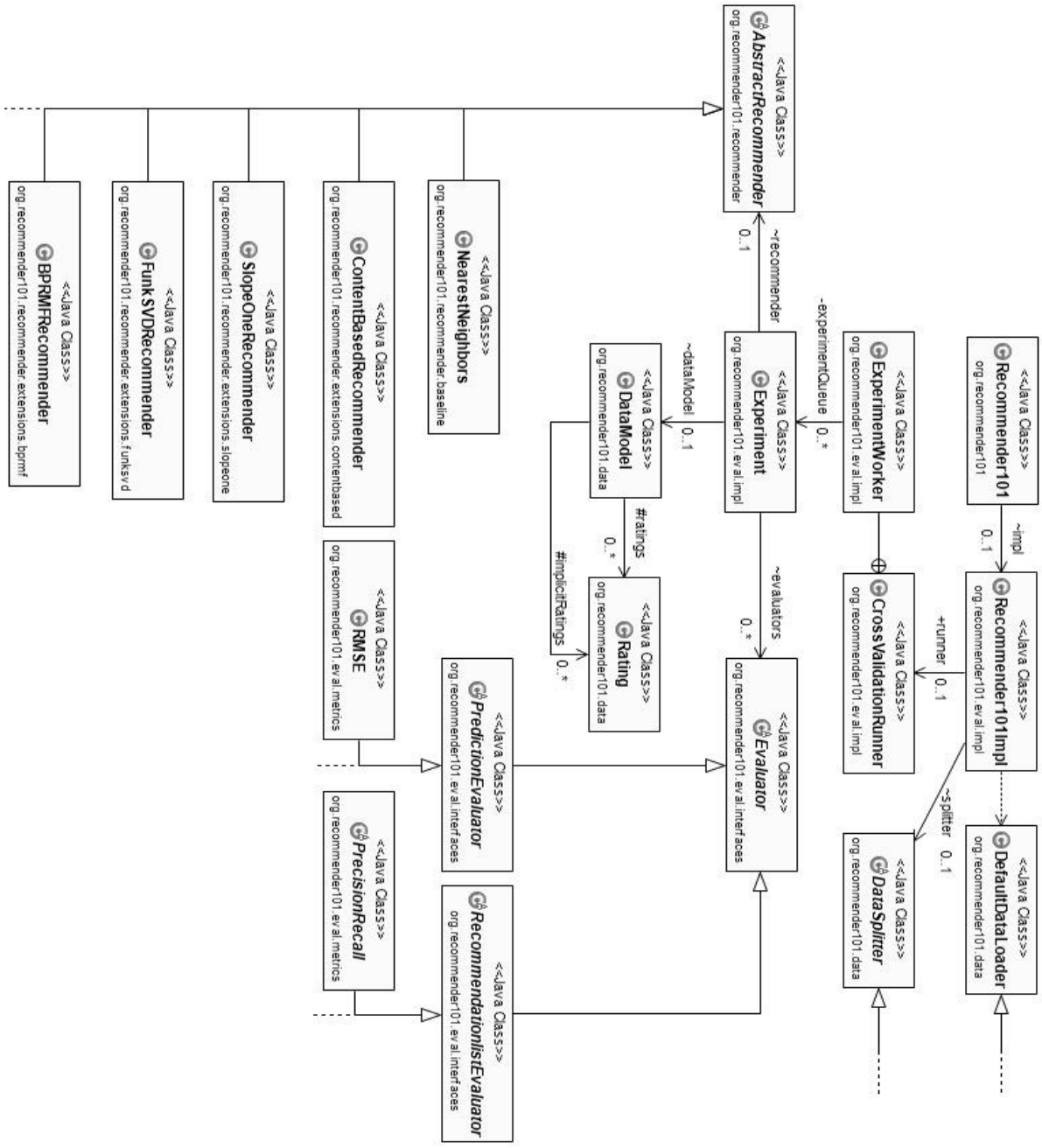

Figure 1. Recommender101 Design [5]

3.1 Explanations

We define explanations as a set of details that accompany the output result, thus making it clear to the researchers what needs to be included in a research output. In Recommender101 a number of settings and parameters are available in the configuration file (recommender101.properties under the conf directory). If these parameters are not properly mentioned by researchers in their work, then the output result could vary significantly [2].

3.2 The proposed approach

In the proposed approach we:

1. Retrieve information from the configuration file.
2. Write the information in the log file along with the evaluation result and explain what this is.

The settings retrieved from the configuration file are the following and are presented in the same way that are saved in the log file:

1. The configuration parameters and settings can be set at the configuration file recommender101.properties that can be found under the conf directory of Recommender101
2. The filename of the dataset is (name of the file goes here)
3. The minimum number of ratings per user to be considered is (number)
4. The minimum number of ratings per considered item is (number)
5. This experiment has used all users, OR This experiment has used (number) users
6. The minimum rating value applied is (number e.g. between 1 to 5)
7. The maximum rating value applied is (number e.g. between 1 to 5)
8. This experiment is based on a (number e.g. 5 or 10) cross fold validation OR this experiment is based on a training/test approach using (number %) for training and (number %) for testing
9. The number of nearest neighbours used is (number)
10. The algorithm used is (name)
11. The metrics used for this experiment are (This is already implemented in recommender101)
12. The results are (This is already implemented in recommender101)

The 12 settings including the evaluation result (12^{th} explanation) as shown above are the explanations saved in the log file and printed on the screen at the end of the evaluation process. The text before the parenthesis is the default explanation style followed by the details that are passed as parameters but saved and printed without parenthesis.

4. Experimental evaluation

The experimental evaluation has been based on the MovieLens 1 million dataset [13], which consists of 6040 users, 4000 movies and 1 million ratings on a 1-5 scale. Furthermore, a 5-fold cross validation approach has been used along with the Recommender101 library [5]. In our previous conference paper of this work, we have shown that explanations can be useful when someone else wants to reproduce experiments [1]. However, simple recommendation approaches such as CF might be easier to replicate compared to more complicated algorithms. For the experiments we have modified the Recommender101 library to take into consideration other variables before a recommendation algorithm is selected to solve the filter bubble problem [14]. A problem, when a user ‘lives’ within a bubble is that results from popular or the same items being recommended again and again on websites, thus leaving outside many items that could potentially interest users. However, such modifications are difficult to replicate, and explanations become a useful component of such a system. We have made the following changes to the library that can be employed before a specific recommendation algorithm is applied and is explained in section 4.1.

4.1 Filter bubble algorithm

This algorithm is applied before a recommendation algorithm is executed, thus making it easier for any recommendation algorithm to be used but also take into consideration the filter bubble problem.

Recommender101Impl: This is a class that initiates the application and reads all experiment parameters from a configuration file. The class has been modified to include five more settings: (1) My_User_count, (2) My_penalty_multiplier, (3) My_Relat_Function, (4) My_fuzzy_norm and (5) My_fuzzy_mv. An explanation of each of these 5 settings follows.

1. **My_User_count:** Takes a value from zero, up to the total number of users found in the dataset. If the value is <=0 then the filter bubble algorithm is skipped and Recommender101 runs as usual.
2. **My_penalty_multiplier:** This is a similarity value between two users. For example, assuming that two users i and z have 5 common ratings with the values of the i user {3,4,2,5,1} and for z {3,2,2,5,2} with a penalty multiplier of 1 then the value returned is (5-1*|3-3|)+(5-1*|4-2|)+(5-1*|2-2|)+(5-1*|5-5|)+(5-1*|1-2|)=22 and in the case where the penalty multiplier is 2 then the value returned is (5-2*|3-3|)+(5-2*|4-2|)+(5-2*|2-2|)+(5-2*|5-5|)+(5-2*|1-2|)=19, which means less similar users. The higher the value, the higher the penalty between two users, thus higher the resulting dissimilarity.
3. **My_Relat_Function:** Since in a dataset many users might have submitted hundreds of ratings and other users very few there might be huge differences when the penalty multiplier is applied. Therefore, to overcome this issue we employ a relativity equation.
4. **My_fuzzy_norm:** This is an equation that calculates an overall similarity value of a user i with all other users.
5. **My_fuzzy_mv:** After the fuzzy norm is calculated then this is the threshold that is used as a decision-making point for which users are similar and which are not.

DefaultDataLoader (loadData): This is another class of Recommender101 that has been modified. In particular the loadData method has been modified to apply the recommendation algorithm after the execution of the Recommender101Impl class.

The steps of the algorithm are the following:

1. Read from the configuration file the My_User_count, thus read the set of users U={U_1, U_2, U_3, …, U_x}, the set of items UM_i = {(U_i,M_1), (U_i,M_2), …,(U_i,M_y)} and the ratings for each item UR_i={(M_1,R), (M_2,R), …,(M_y,R) }.
2. Find all common items for each user U_x with all other users of the dataset U. Then, for each user i create a set X with common items, thus creating X^2 sets as shown in equation 3.

$$\forall\ (U_i\ , U_z\ \in U)\ , C_{i,z} = UM_i \cap UM_z \tag{3}$$

3. Apply equation 4 to calculate the matching-degree between two users i and z. In this equation y is the number of common ratings (length of $C_{i,z}$), $U_i M_k R$ is the rating of user I for an item k and $U_z M_k R$ is the rating of user z for the same item k.

$$\forall\ C_{i,z}\ , MD_{i,z} = \sum_{k=1}^{y} (MaxRating - PM * |U_i M_k R - U_z M_k R|). \tag{4}$$

4. Step 3 returns an X * X table with matching-degree values and this step will return and Min value, a Max value, and Average (avg) value and a standard deviation (SD) value.

5. Calculate variables F and G which will be used for the defuzzification process of the matching-degree for every user combination U_i, z. The calculation of F is shown in equation 5 and of G in equation 6.

$$F = \begin{cases} Avg - (2 * SD), & if\ (Avg - 2(SD) \geq \text{Min}) \\ Min, & if\ (Avg - 2(SD) < \text{Min}) \end{cases} \quad (5)$$

$$G = \begin{cases} Avg + (2 * SD), & if\ (Avg - 2(SD) \geq \text{Max}) \\ Max, & if\ (Avg - 2(SD) < \text{Max}) \end{cases} \quad (6)$$

6. The defuzzification will take place of all $C_{i,z}$ values based on two fuzzy sets Low and high using semi-trapezoid function as shown in figure 2.

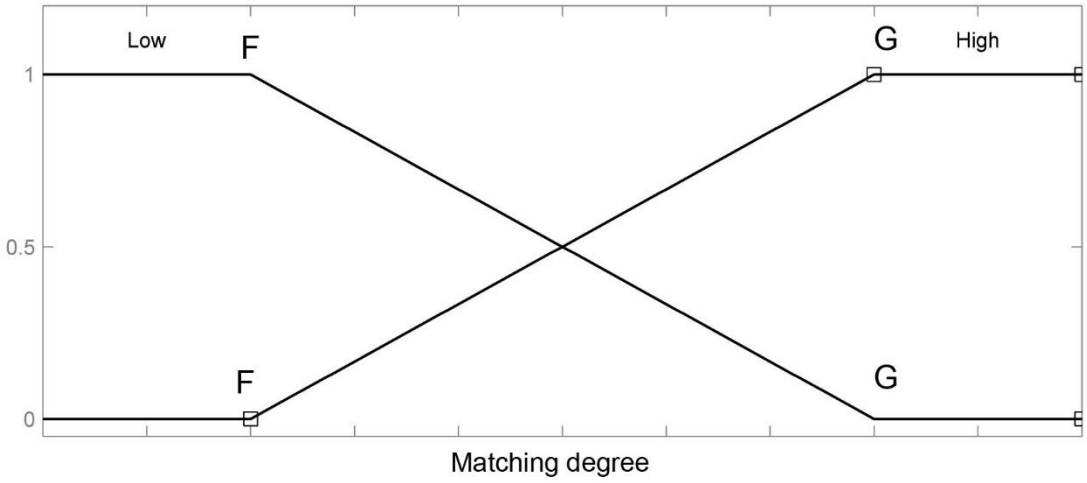

Figure 2. Filter Bubble Algorithm Defuzzification process

Then the low and high similarity values are calculated as shown in equations 7 and 8 respectively.

$$LowC_{i,z}(C_{i,z}; F, G) = \begin{cases} 1, C_{i,z} \leq F \\ \dfrac{G - C_{i,z}}{G - F}, F < C_{i,z} < G \\ 0, C_{i,z} \geq G \end{cases} \quad (7)$$

$$HighC_{i,z}(C_{i,z}; F, G) = \begin{cases} 0, C_{i,z} \leq F \\ \dfrac{C_{i,z} - F}{G - F}, F < C_{i,z} < G \\ 1, C_{i,z} \geq G \end{cases} \quad (8)$$

7. At this step the final user matching degree is calculated based on the high fuzzy set through sigma count average.
8. For each user the final value is checked and if it is smaller than the My_fuzzy_mv value do not pass this user to the recommendation algorithm.
9. Start Recommender101 based on the settings and algorithms found in the configuration file.

4.2 Evaluation metrics

For the evaluation, well-known metrics have been used. These include MAE which has already been defined in equation 2 in the background section, Root Mean Square Error (RMSE), Precision, Recall and Mean Average Precision. All these metrics have widely been used in the literature [1,2,3,4,5]. MAE and RMSE are rating error prediction metrics and lower values represent better predictions, whereas Precision, Recall and MAP are information retrieval metrics and represent the quality of the retrieved recommendations and higher values are better. Precision represents the number of correct recommendations within a list of top-N recommendations, Recall is the fraction of relevant recommendations successfully retrieved and MAP is the percentage of correct recommendations appearing on top of the recommendation list. RMSE is defined in equation 9 and as in MAE p_i is the predicted rating and r_i is the actual rating in the summation. Precision is defined in equation 10, Recall in equation 11 and MAP in equation 12, where Q is the set of queries and AP is the average precision of a query q. For the information retrieval metrics, the top-10 recommendations are requested in each evaluation scenario.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)} \tag{9}$$

$$precision = \frac{Correctly\ recommended\ items}{Total\ recommender\ items} \tag{10}$$

$$recall = \frac{Correctly\ recommended\ items}{Relevant\ items} \tag{11}$$

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{12}$$

4.3 Results

We performed four different evaluation cycles, with the last three cycles performed by a different researcher without using the explanations provided from the first evaluation cycle. The aim of this was to evaluate the number of cycles required to reproduce an experiment and the results are shown in tables 3, 4, 5 and 6 respectively. The algorithms used are FunkSVD and SlopeOne which are available within Recommender101 and the number of selected users represents the first users as found in the dataset. For example, when mentioned 1000 users processed it refers to the first 1000 users of the dataset. For the results presented in table 3 the following settings have been used: users to proceed 1000, penalty multiplier 2.0, Relativity function AVG, final fuzzy calculation function SCA and fuzzy mv threshold 0.5. In table 4 the settings have been set to: users to proceed 1000, penalty multiplier 1.6, relativity function SUM, final fuzzy calculation function SCA and fuzzy mv threshold 0.3. For the results in table 5 the following settings have been used: users to proceed 2000, penalty multiplier 1.8, relativity function SUM, final fuzzy calculation SCA and fuzzy mv threshold 0.3. For the fourth evaluation cycles results shown in table 6 the following settings have been used: users to proceed 4000, penalty multiplier 1.7, relativity function SUM, final fuzzy calculation SCA and fuzzy mv threshold 0.32. The results show that significant differences exist between the evaluation cycles. These can reach a 9.2% difference in terms of precision, which can clearly affect the selection of items included in the top-N recommendation lists for most users.

Metric	FunkSVD	SlopeOne
MAE	0.670	0.673
RMSE	0.853	0.859
Precision	0.739	0.740
Recall	0.522	0.526
MAP	0.795	0.794

Table 3. First evaluation cycle

Metric	FunkSVD	SlopeOne
MAE	0.690	0.701
RMSE	0.879	0.892
Precision	0.806	0.786
Recall	0.281	0.275
MAP	0.844	0.817

Table 4. Second evaluation cycle

Metric	FunkSVD	SlopeOne
MAE	0.711	0.722
RMSE	0.907	0.918
Precision	0.802	0.796
Recall	0.278	0.277
MAP	0.834	0.822

Table 5. Third evaluation cycle

Metric	FunkSVD	SlopeOne
MAE	0.713	0.722
RMSE	0.909	0.918
Precision	0.774	0.769
Recall	0.341	0.340
MAP	0.815	0.804

Table 6. Fourth evaluation cycle

5. Discussion

Reproducibility of experiments in recommender systems evaluation is an issue that needs to be addressed effectively. In this paper we have proposed an approach that is based on explanations. This aims to assist researchers towards reproducing experiments under certain conditions. Initially, we have shown that straightforward algorithms like CF can be easily reproduced. However, in the evaluation section we have developed a pre-processing algorithm which when introduced at the first stage of the experiment, can solve the filter bubble problem and also provide quality recommendations by taking into consideration a number of parameters. Following this any recommendation algorithm can be used. By developing this algorithm, we want to explain to researchers when parameters change and where these details might not be available, significant differences exist between lists of recommendations. For example, in FunkSVD results at table 3 Precision and MAP which measure the quality of the list of the top-N recommendation provided 73.9% for Precision and 79.5% for MAP whereas in table 4 the same algorithm provided 80.6% for Precision and 84.4% for MAP. This shows a 9.2% difference in Precision and 6.2% difference for MAP, which is substantial. Similar differences exist for the SlopeOne algorithm as well and for all other metrics in all tables.

The experiments took place in a controlled environment where the values of the pre-processing algorithms were assumed to be unknown for tables 4, 5 and 6. Without the use of explanation a second researcher was asked to replicate the results of table 3 and following a number of evaluation cycles the possibility of reproducing the results of table 3 was assessed. After three evaluation cycles, the results of table 6 appear to be closer to the results of table 3 but the differences are still minor but noticeable. It is difficult however to exactly reproduce an experiment, nevertheless it is shown that many evaluation cycles are often necessary to achieve a result that is close to the one that needs to be reproduced. This problematic process known as 'experimental evaluation' in recommender systems can be easily overcome with the use of explanations that can supply the results with an output that explains all settings and parameters. Thus, the effort and time required from researchers for the evaluation of other algorithms is reduced and they are allowed to concentrate on researching and developing of their own ideas and methods. Researchers will often compare their proposed algorithm with a number of different state-of-the-art alternatives, and it will take time to try and guess settings until an optimal combination has been achieved. As shown in the results a few cycles were required to achieve a result that it was relatively close to the original one, but what if there were more settings and more algorithms or if settings were unavailable.

The proposed approach has been extended to include the use of the new variables and the output includes a total of 17 settings. The 12 mentioned in section 3 and the 5 mentioned at the beginning of section 4 (My_User_count, My_penalty_multiplier, My_Relat_Function, My_fuzzy_norm and My_fuzzy_mv). The broad idea of explanations is to assist researchers to reproduce experiments, since there might be cases where (a) of algorithms very difficult to understand, (b) settings not mentioned and (c) algorithmic aspects not mentioned. In such cases it is very useful to reduce the evaluation cycles and time and gain an advantage by using this time for further research and development.

Recommender systems are not the only area of Artificial Intelligence that could benefit from enhanced reproducibility. Is through an area that, given the plethora of data that we have accumulated and the need for enhanced decision support in a range of fields, could prove very important in application areas such as medical systems and systems that control autonomous vehicles. In both these areas the accuracy of the recommendation results is absolutely essential, the need for reproducing experiments is a fundamental requirement and the use of explanations can make this process easier and more efficient.

6. Conclusions and future work

In this paper we highlighted the problem of reproducibility in recommender systems' evaluation. Although, it was shown in previous research that it is difficult to reproduce results using different offline evaluation libraries, the reproducibility of results becomes achievable if the correct settings and parameters are used within the same library. Thus, we have proposed an approach that is based on explanations that can be used to assist researchers in reproducing the results of an experimental

evaluation. The initial evaluation results are promising and can assist towards further development in this direction. Our approach can be straightforwardly implemented by researchers using other libraries. Furthermore, in our future work we aim to provide a visualized approach of the explanations and explore the application of explanation in different domains such as medical systems and autonomous vehicles.

Conflict of Interest: The authors declare that they have no conflict of interest.

References

1. Polatidis, N., & Pimenidis, E. (2018, September). Reproduction of experiments in recommender systems evaluation based on explanations. In *International Conference on Engineering Applications of Neural Networks* (pp. 194-200). Springer, Cham.

2. Polatidis, N., Kapetanakis, S., Pimenidis, E., & Kosmidis, K. Reproducibility of experiments in recommender systems evaluation 14th International Conference on Artificial Intelligence Applications and Innovations, AIAI 2018. IFIP AICT 519 pp.

3. Said, A., Bellogín, A.: Comparative recommender system evaluation. Proc. 8th ACM Conf. Recomm. Syst. - RecSys '14. 129–136 (2014).

4. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22, 5–53 (2004).

5. Jannach, D., Lerche, L., Gedikli, F., Bonnin, G.: What recommenders recommend–an analysis of accuracy, popularity, and sales diversity effects. In: User Modeling, Adaptation, and Personalization. pp. 1–13 (2013).

6. Konstan, J.A., Adomavicius, G.: Toward Identification and Adoption of Best Practices in Algorithmic Recommender Systems Research. In: Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation. pp. 23–28. ACM, New York, NY, USA (2013).

7. Bellogin, A., Castells, P., Said, A., Tikk, D.: Workshop on reproducibility and replication in recommender systems evaluation. In: Proceedings of the 7th ACM conference on Recommender systems - RecSys '13. pp. 485–486 (2013).

8. Bellogin, A., Castells, P., Said, A., Tikk, D.: Report on the Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys). SIGIR Forum. 48, 29–35 (2014).

9. Košir, A., Odić, A., Tkalčič, M.: How to improve the statistical power of the 10-fold cross validation scheme in recommender systems. In: RecSys RepSys 2013: Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation. pp. 3–6 (2013).

10. Said, A., Bellogín, A.: RiVal – A Toolkit to Foster Reproducibility in Recommender System Evaluation. RecSys 2014 Proc. 8th ACM Conf. Recomm. Syst. 371–372 (2014).

11. Hernández del Olmo, F., Gaudioso, E.: Evaluation of recommender systems: A new approach. Expert Syst. Appl. 35, 790–804 (2008).

12. Peker, S., Kocyigit, A.: mRHR: A Modified Reciprocal Hit Rank Metric for Ranking Evaluation of Multiple Preferences in Top-N Recommender Systems. In: Dichev, C. and Agre, G. (eds.) Artificial Intelligence: Methodology, Systems, and Applications: 17th International Conference, AIMSA 2016, Varna, Bulgaria, September 7-10, 2016, Proceedings. pp. 320–329. Springer

International Publishing, Cham (2016).

13. Harper, F.M., Konstan, J.A.: The MovieLens Datasets. ACM Trans. Interact. Intell. Syst. 5, 1–19 (2015).

14. Pariser, E. (2011). *The filter bubble: What the Internet is hiding from you*. Penguin UK.