# Hybrid data set optimization in recommender systems using Fuzzy T-norms

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

[1,3] School of Engineering, Department of Civil Engineering, Faculty of Mathematics Programing and General courses, Democritus University of Thrace, Kimmeria, Xanthi, Greece
papaleon@sch.gr[1], liliadis@civil.duth.gr[3]
[2] Department of Computer Science and Creative Technologies, University of the West of England, BS16 1QY, Bristol, United Kingdom
Elias.Pimenidis@uwe.ac.uk

**Abstract.** A recommender system uses specific algorithms and techniques in order to suggest specific services, goods or other type of recommendations that users could be interested in. User's preferences or ratings are used as inputs and top-N recommendations are produced by the system. The evaluation of the recommendations is usually based on accuracy metrics such as the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE), while on the other hand Precision and Recall is used to measure the quality of the top-N recommendations. Recommender systems development has been mainly focused in the development of new recommendation algorithms. However, one of the major problems in modern offline recommendation system is the sparsity of the datasets and the selection of the suitable users Y that could produce the best recommendations for users X. In this paper, we propose an algorithm that uses Fuzzy sets and Fuzzy norms in order to evaluate the correlation between users in the data set so the system can select and use only the most relevant users. At the same time, we are extending our previous work about Reproduction of experiments in recommender systems by developing new explanations and variables for the proposed new algorithm. Our proposed approach has been experimentally evaluated using a real dataset and the results show that it is really efficient and it can increase both accuracy and quality of recommendations.

**Keywords:** Recommender systems, Evaluation, Explanations, Reproducibility, Fuzzy logic, T-Norms

## 1 Introduction

The use of recommender systems is very common, especially in applications like e-Commerce and social networks. Products recommendation techniques can reduce the overall searching time of an e-shop user and increase sales. Apart from e-commerce, recommendation technology is also used in various other less known domains such as music recommendation [1, 4], books [2], documents [3], television programs [5], people to people recommendation [1], applications in markets [6], e-learning [7] and Web search [8].

The increasingly importance, 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. Researchers are mostly focusing in creating more effective algorithms and models by trying to minimize the MAE and RMSE while at the same time they are trying to improve precision and recall of top-N recommendations [9, 10]. While this is important to do, it should be also noted that the problem of reproducing the results exists and it is considered important [11]. In a previous work of the research team [13], offline recommender system results were successfully reproduced using an explanation-based approach.

There are different libraries that can be used for developing and testing a recommendation algorithm and include, among others, Recommender101, Apache Mahout, LensKit and MyMediaLite [13]. Polatidis et al [1], 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.

In this paper we are further expanding our previous work by extending Recommender101 libraries, modifying the methodology that Recommender101 is selecting users with common ratings and adding all the necessary parameters in the configuration files of Recommender101.

The rest of the paper is organized as follows: Section 2 provides the relevant background and describes related work, section 3 delivers the proposed extension and algorithm of the team's previous method, section 4 describes the experiments, the results and the discussion while conclusions and future work are included in section 5.

## 2 Related work

Recommenders are often evaluated and compared offline using datasets collected from online platforms [18]. Evaluation can be done by 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 [1][19].

Research papers that propose new recommendation algorithms will typically describe the experimental setup, the dataset used, and the framework that was [13]. A major challenge in recommender system evaluation is that there are many different libraries for evaluating algorithms and the possibility of having one single library or making all the current libraries following a universal or standardized approach is rather impossible [1].

The idea of a unified approach which can facilitate a common reference baseline for recommendation experiments across different frameworks and a set of guidelines to tackle a cross-industry challenge was proposed in [1] and was the trigger point for the proposal and development of a reproducibility framework that combines the ability to reproduce recommendation experiments as well as the support of new algorithms and methodologies.

### 2.1 Reproduction of experiments in recommender systems evaluation based on explanations

In [13], the problem of reproducibility in recommender systems evaluation was highlighted and the importance of the correct settings and parameters which were used within the library was indicated.

In the proposed approach we:

1. Retrieve information from the configuration file
2. Write the information in the log file along with 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 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 neighbors 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 initial evaluation results were promising and triggered the idea of extending the configuration file and the necessary libraries of recommender101in order to achieve a better way of selecting the most proper users Y that could produce the best recommendations for users X.

### 2.2 Reproducibility of recommender systems experiments based on explanations

On [13] extension we modified the source code of Recommender101.java, Recommender101Impl.java and DefaultDataLoader.java in order to:

- Enrich the recommender101 configuration file (recommender101.properties) with new parameters that will help us in optimizing the use of data set.

- Improve both accuracy and quality of recommendations by providing tools to select the most proper users Y that could produce the best recommendations for users X
- Strengthen our previous work by checking the offline reproduction of recommender system experiment with the use of modified source code and extended configuration file.

Five new parameters have been incorporated in the configuration file of Recommender101and also were defined in the source code of the platform. Those parameters are:

1. **My_User_count**: Takes a value from zero, up to the total number of users found in the dataset. It works in co-operation with the already existing parameter sampleNUsers and it should be grater than sampleNUsers. It indicates the total number of users which will be used in order to select the best matches for sampleNUsers. If the value is <=0 then the modified algorithm is skipped and Recommender101 runs as usual.
2. **My_penalty_multiplier**: This is a similarity value between two users. The higher the penalty multiplier the lower similarity between users with no identical ratings. 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.
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, AVG or SUM can be used as parameters in My_Relat_Function. AVG will normalize the differences and SUM will keep the raw calculated values.
4. **My_fuzzy_norm**: This parameter represents the equation that calculates the overall similarity value of a user i with all other users. Sigma Count Average [14] was used in the basic implementation and in this paper Hamacher and Einstein product are introduced and used.
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.

By default, Recommender101 system uses a partial set of the available users in the data set in order to calculate top-N recommendations. Already existing sampleNUsers parameter is used to define the size of that partial set of users that will be used. The proposed algorithm uses the similarity between My_User_count number of users in order to select the best ones that will be included in the data set.

## 3 Proposed extension and Algorithm

DefaultDataLoader is the class of Recommender101 that is used to read the dataset that is described in the configuration file of the platform. We decided not to modify the recommendation algorithm but to work in DefaultDataLoader class. By doing that the main recommendation algorithm works on the same way and we can apply any type of source dataset manipulation without the need of rechecking the recommendation methodology. The core class of the platform is getting the new, processed, dataset and the main algorithm and the evaluation metrics remain the same.

The steps of the algorithm are the following:

1. Read from the configuration file the My_User_count, thus read the set of users $U=\{U_1, U_2, U_3, \ldots, U_x\}$, the set of items $UM_i = \{(U_i,M_1), (U_i,M_2), \ldots,(U_i,M_y)\}$ and the ratings for each item $UR_i=\{(M_1,R), (M_2,R), \ldots,(M_y,R)\}$.
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 1.
$$\forall (U_i, U_z \in U), C_{i,z} = UM_i \cap UM_z \quad (1)$$
3. Apply equation 2 to calculate the matching-degree between two users i and z. In this equation y is the number of common ratings (length of $C_{i,z}$), $U_iM_kR$ is the rating of user I for an item k and $U_zM_kR$ is the rating of user z for the same item k.
$$\forall C_{i,z}, MD_{i,z} = \sum_{k=1}^{y} (MaxRating - PM * |U_iM_kR - U_zM_kR|). \quad (2)$$
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 fuzzification process of the matching-degree for every user combination $U_i, z$. The calculation of F is shown in equation 3 and of G in equation 4.
$$F = \begin{cases} Avg - (2 * SD), & if\ (Avg - 2(SD) \geq \text{Min}) \\ Min, & if\ (Avg - 2(SD) < \text{Min}) \end{cases} \quad (3)$$
$$G = \begin{cases} Avg + (2 * SD), & if\ (Avg - 2(SD) \geq \text{Max}) \\ Max, & if\ (Avg - 2(SD) < \text{Max}) \end{cases} \quad (4)$$
6. The fuzzification will take place of all $C_{i,z}$ values based on two fuzzy sets Low and High using two symmetric semi-trapezoid function as shown in equations 5 and 6.

$$LowC_{i,z}\left(C_{i,z};F,G\right)=\begin{cases}1, C_{i,z}\le F\\ \frac{G-C_{i,z}}{G-F}, F<C_{i,z}<G\\ 0, C_{i,z}\ge G\end{cases} \quad (5)$$

$$HighC_{i,z}\left(C_{i,z};F,G\right)=\begin{cases}0, C_{i,z}\le F\\ \frac{C_{i,z}-F}{G-F}, F<C_{i,z}<G\\ 1, C_{i,z}\ge G\end{cases} \quad (6)$$

7. The final user matching degree is calculated based on the high fuzzy set through Sigma Count average, hybrid Hamacher or Einstein products.
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.

### 3.1 Hybrid Hamacher and Einstein products

Hamacher and Einstein products are two well know Fuzzy T-Norms that are widely used in Intuitionistic Fuzzy Information Aggregation [15][16] and they are calculated based on the equations 7 (Hamacher product) and 8 (Einstein product).

$$\bar{A}\cap\bar{B}=\frac{\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)}{\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)} \quad (7)$$

$$\bar{A}\cap\bar{B}=\frac{\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)}{2-[\mu_{\bar{A}}(x)+\mu_{\bar{B}}(x)-\mu_{\bar{A}}(x)\mu_{\bar{B}}(x)]} \quad (8)$$

In both cases membership values equals to Zero will result both products to be equal to zero. In that case relativity of the user will be also zeroed so in our hybrid version of Hamacher and Einstein products, zero membership values are not used in the calculations. This modification can be adopted due to the existence of custom variable My_Relat_Function which will "promote" users with high number of ratings even if they got some zeros in membership values.

After calculating the desired product for each user, the values are linearly normalized in [0,1] in order to be in match with My_fuzzy_mv threshold.

## 4 Experiments and results

The proposed methodology was extensively tested by measuring the performance and the accuracy of the Recommender101 under different configuration schemas.

At first part of testing we used the system with the default configuration and without processing the dataset. Four different values of sampleNUsers (100, 400, 800 and 1600) were used.

At the second part of testing we used the system by enabling the source dataset processing with different values for My_User.count and My_fuzzy.norm custom variables. Once again, each case was tested with the same four different values of sampleNUsers as before. The value of My_User.count valiable was set to 300, 1200, 2400, 4600 users accordingly to sampleNUsers. Penalty multiplier was set to 1.7, Fuzzy_mv threshold was set to 0.5 and realativity function was set to AVG. All three alternatives for My_fuzzy.norm, Sigma Count average, hybrid Hamacher or Einstein products, were also used.

In total 16 different configurations were used, and the same number of results were created. In all cases both "FunkSVDRecommender" and "SlopeOneRecommender" algorithms were used.

Custom variable's values were set based on the fact that this paper focuses in the importance of source dataset prospecting and not in finding the best values for achieving the maximum accuracy.

MovieLens 1 million dataset [17], which consists of 6040 users, 4000 movies and 1 million ratings in a 1-5 scale was used for the experimental evaluation of the method.

All experiment took place on a Windows 10 64bit computer with 64GB ram and an i5-4570 cpu.

### 4.1 Evaluation metrics

The evaluation was based on well-known and widely used metrics like Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) for error rating, and Precision and Recall for measuring the quality of the top-N recommendations [1,11,13].

MAE and RMSE are rating error prediction metrics and lower values represent better predictions, whereas Precision and Recall are information retrieval metrics and represent the quality of the retrieved recommendations and higher values are better.

MAE, RMSE, precision and recall equations are following.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |pi - ri| \quad (9)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (p_i - r_i)} \quad (10)$$

$$precision = \frac{Correctly\ recommended\ items}{Total\ recommender\ items} \quad (11)$$

$$recall = \frac{Correctly\ recommended\ items}{Relevant\ items} \quad (12)$$

Execution Duration is measured in seconds and was used for performance evaluation of the proposed method.

### 4.2 Experimental results

First four sets of results, table 1, refer to first run of the system with the default Recommender101 configuration, with no dataset process and values of sampleNUsers equal to 100, 400, 800 and 1600.

**Table 1.** Default Recommender101 execution.

| ***sampleNUsers*** | ***100*** | ***400*** | ***800*** | ***1600*** |
|---|---|---|---|---|
| ***Items*** | 3706 | 3706 | 3706 | 3706 |
| ***Ratings*** | 14720 | 60796 | 137742 | 274529 |
| ***Sparsity*** | 0,04 | 0,041 | 0,046 | 0,046 |
| ***Avg. Ratings/User*** | 147,2 | 151,99 | 172,178 | 171,581 |
| ***Avg. Ratings/Item*** | 3,972 | 16,405 | 37,167 | 74,077 |
| ***Max /User*** | 1050 | 1277 | 1302 | 1743 |
| ***Max /item*** | 59 | 219 | 469 | 917 |
| ***MAE FunkSVD*** | 0,802 | 0,739 | 0,725 | 0,72 |
| ***MAE SlopeOne*** | 0,792 | 0,733 | 0,711 | 0,692 |
| ***RMSE FunkSVD*** | 1,024 | 0,94 | 0,921 | 0,915 |
| ***RMSE SlopeOne*** | 1,004 | 0,933 | 0,904 | 0,882 |
| ***Precision FunkSVD*** | 0,693 | 0,724 | 0,742 | 0,755 |
| ***Precision SlopeOne*** | 0,691 | 0,724 | 0,735 | 0,739 |
| ***Recall FunkSVD*** | 0,488 | 0,514 | 0,518 | 0,523 |
| ***Recall SlopeOne*** | 0,483 | 0,517 | 0,509 | 0,509 |
| ***Execution Duration*** | 22 | 133 | 294 | 632 |

Next results, table 2, are showing the use of the proposed method with Sigma Count average as the final fuzzification method.

**Table 2.** Sigma Count average as final fuzzification method results.

| ***sampleNUsers / My_User.count*** | ***100 (300)*** | ***400 (1200)*** | ***800 (2400)*** | ***1600 (4800)*** |
|---|---|---|---|---|
| ***Items*** | 2972 | 3418 | 3576 | 3662 |
| ***Ratings*** | 13137 | 67405 | 134833 | 276467 |
| ***Sparsity*** | 0,044 | 0,049 | 0,047 | 0,047 |
| ***Avg. Ratings/User*** | 131,37 | 168,512 | 168,541 | 172,792 |
| ***Avg. Ratings/Item*** | 4,42 | 19,721 | 37,705 | 75,496 |
| ***Max /User*** | 608 | 1518 | 1323 | 1271 |
| ***Max /item*** | 61 | 255 | 514 | 961 |
| ***MAE FunkSVD*** | 0,773 | 0,717 | 0,705 | 0,707 |
| ***MAE SlopeOne*** | 0,765 | 0,713 | 0,692 | 0,682 |
| ***RMSE FunkSVD*** | 0,991 | 0,912 | 0,894 | 0,898 |
| ***RMSE SlopeOne*** | 0,972 | 0,917 | 0,879 | 0,868 |
| ***Precision FunkSVD*** | 0,714 | 0,737 | 0,751 | 0,763 |
| ***Precision SlopeOne*** | 0,71 | 0,736 | 0,744 | 0,744 |

| | | | | |
|---|---|---|---|---|
| ***Recall FunkSVD*** | 0,519 | 0,517 | 0,527 | 0,524 |
| ***Recall SlopeOne*** | 0,504 | 0,519 | 0,516 | 0,512 |
| ***Execution Duration*** | 15 | 136 | 265 | 385 |

Hybrid Hamacher product results are in table 3 below.

**Table 3.** Hybrid Hamacher product as final fuzzification method results.

| ***sampleNUsers / My_User.count*** | ***100 (300)*** | ***400 (1200)*** | ***800 (2400)*** | ***1600 (4800)*** |
|---|---|---|---|---|
| ***Items*** | 2931 | 3375 | 3507 | 3642 |
| ***Ratings*** | 12997 | 67302 | 135107 | 278365 |
| ***Sparsity*** | 0,044 | 0,049 | 0,048 | 0,048 |
| ***Avg. Ratings/User*** | 131,37 | 168,512 | 168,541 | 172,792 |
| ***Avg. Ratings/Item*** | 4,46 | 19,714 | 37,788 | 79,14 |
| ***Max /User*** | 685 | 1571 | 1334 | 1420 |
| ***Max /item*** | 68 | 276 | 603 | 1007 |
| ***MAE FunkSVD*** | 0,761 | 0,711 | 0,699 | 0,703 |
| ***MAE SlopeOne*** | 0,759 | 0,711 | 0,689 | 0,679 |
| ***RMSE FunkSVD*** | 0,981 | 0,91 | 0,899 | 0,897 |
| ***RMSE SlopeOne*** | 0,972 | 0,918 | 0,861 | 0,86 |
| ***Precision FunkSVD*** | 0,723 | 0,74 | 0,767 | 0,766 |
| ***Precision SlopeOne*** | 0,723 | 0,736 | 0,741 | 0,748 |
| ***Recall FunkSVD*** | 0,52 | 0,518 | 0,531 | 0,528 |
| ***Recall SlopeOne*** | 0,512 | 0,519 | 0,525 | 0,519 |
| ***Execution Duration*** | 16 | 142 | 301 | 410 |

Hybrid Einstein product results are in table 4 below.

**Table 4.** Hybrid Einstein product as final fuzzification method results.

| ***sampleNUsers / My_User.count*** | ***100 (300)*** | ***400 (1200)*** | ***800 (2400)*** | ***1600 (4800)*** |
|---|---|---|---|---|
| ***Items*** | 2972 | 3418 | 3576 | 3662 |
| ***Ratings*** | 13137 | 67405 | 134833 | 276467 |
| ***Sparsity*** | 0,044 | 0,049 | 0,047 | 0,047 |
| ***Avg. Ratings/User*** | 131,37 | 168,512 | 168,541 | 172,792 |
| ***Avg. Ratings/Item*** | 4,42 | 19,721 | 37,705 | 75,496 |
| ***Max /User*** | 680 | 1603 | 1371 | 1407 |
| ***Max /item*** | 72 | 268 | 530 | 980 |
| ***MAE FunkSVD*** | 0,76 | 0,714 | 0,701 | 0,69 |
| ***MAE SlopeOne*** | 0,756 | 0,709 | 0,671 | 0,677 |
| ***RMSE FunkSVD*** | 0,983 | 0,904 | 0,89 | 0,891 |
| ***RMSE SlopeOne*** | 0,972 | 0,917 | 0,879 | 0,868 |
| ***Precision FunkSVD*** | 0,722 | 0,739 | 0,761 | 0,764 |
| ***Precision SlopeOne*** | 0,717 | 0,738 | 0,748 | 0,751 |

| | | | | |
|---|---|---|---|---|
| ***Recall FunkSVD*** | 0,523 | 0,519 | 0,529 | 0,531 |
| ***Recall SlopeOne*** | 0,511 | 0,52 | 0,531 | 0,527 |
| ***Execution Duration*** | 16 | 142 | 298 | 409 |

Next three tables show the percentage difference on basic evaluation metrics between standard Recommender101 execution and Sigma Count, Hybrid Hamacher product and Hybrid Einstein product.

**Table 5.** Hybrid Einstein product as final fuzzification method results.

| ***sampleNUsers / My_User.count*** | ***100 (300)*** | ***400 (1200)*** | ***800 (2400)*** | ***1600 (4800)*** |
|---|---|---|---|---|
| ***MAE FunkSVD*** | -3,62% | -2,98% | -2,76% | -1,81% |
| ***MAE SlopeOne*** | -3,41% | -2,73% | -2,67% | -1,45% |
| ***RMSE FunkSVD*** | -3,22% | -2,98% | -2,93% | -1,86% |
| ***RMSE SlopeOne*** | -3,19% | -1,71% | -2,77% | -1,59% |
| ***Precision FunkSVD*** | 3,03% | 1,80% | 1,21% | 1,06% |
| ***Precision SlopeOne*** | 2,75% | 1,66% | 1,22% | 0,68% |
| ***Recal FunkSVD*** | 6,35% | 0,58% | 1,74% | 0,08% |
| ***Recal SlopeOne*** | 4,35% | 0,39% | 1,38% | 0,59% |

**Table 6.** Hybrid Einstein product as final fuzzification method results.

| ***sampleNUsers / My_User.count*** | ***100 (300)*** | ***400 (1200)*** | ***800 (2400)*** | ***1600 (4800)*** |
|---|---|---|---|---|
| ***MAE FunkSVD*** | -5,11% | -3,79% | -3,59% | -2,36% |
| ***MAE SlopeOne*** | -4,17% | -3,00% | -3,09% | -1,88% |
| ***RMSE FunkSVD*** | -4,20% | -3,19% | -2,39% | -1,97% |
| ***RMSE SlopeOne*** | -3,19% | -1,61% | -4,76% | -2,49% |
| ***Precision FunkSVD*** | 4,33% | 2,21% | 3,37% | 1,46% |
| ***Precision SlopeOne*** | 4,63% | 1,66% | 0,82% | 1,22% |
| ***Recal FunkSVD*** | 6,56% | 0,78% | 2,51% | 0,96% |
| ***Recal SlopeOne*** | 6,00% | 0,39% | 3,14% | 1,96% |

**Table 7.** Hybrid Einstein product as final fuzzification method results.

| ***sampleNUsers / My_User.count*** | ***100 (300)*** | ***400 (1200)*** | ***800 (2400)*** | ***1600 (4800)*** |
|---|---|---|---|---|
| ***MAE FunkSVD*** | -5,24% | -3,38% | -3,31% | -4,17% |
| ***MAE SlopeOne*** | -4,55% | -3,27% | -5,63% | -2,17% |
| ***RMSE FunkSVD*** | -4,00% | -3,83% | -3,37% | -2,62% |
| ***RMSE SlopeOne*** | -3,19% | -1,71% | -2,77% | -1,59% |
| ***Precision FunkSVD*** | 4,18% | 2,07% | 2,56% | 1,19% |
| ***Precision SlopeOne*** | 3,76% | 1,93% | 1,77% | 1,62% |
| ***Recal FunkSVD*** | 7,17% | 0,97% | 2,12% | 1,53% |
| ***Recal SlopeOne*** | 5,80% | 0,58% | 4,32% | 3,54% |

### 4.3 Discussion

We can see from the tables above that in all the cases that the extended DefaultDataLoader class of Recommender101 was used the recommendation results have been improved. Data sparsity has improved up to 20% (from 0.041 to 0.049) in the experiments where 400 sampleNUsers were provided to Recommender101. MAE has improved by 1.45% to 5.63% and RMSE improved by 1.59% to 4.76%. While checking the quality of recommendations we can see that Precision has improved by 0.68% to 4.63% and Recall by 0.08% to 7,17%.

At the same time, we can see that the execution duration of the recommendation algorithm drops up to 40% (from 632sec to 385 sec) while only in few cases it can rise only up to 6% (from 133 sec to 136 sec).

Although that, as mentioned before, the selection of the optimal values for the new custom valuables is not the main scope of this paper we run an extra experiment by using the Hybrid Einstein product as final fuzzification method and setting the fuzzy membership value (FMV) threshold to 0.55 instead of 0.50. We run the experiment only for the combination of 1600/4800 because in that case we saw that improvement was mainly reduced compared to the other three combinations. The following table 8 shows the results of that extra experiment compared with the results with the default run of Recommender101.

**Table 8.** Comparison of experimental results of run with FMV threshold equal to 0.55 and standard Recommender101 run.

| ***sampleNUsers / My_User.count*** | ***Extra experiment*** | ***Default run*** | ***Difference*** |
|---|---|---|---|
| ***Items*** | 3574 | 3706 | -132 |
| ***Ratings*** | 337632 | 274529 | 63103 |
| ***Sparsity*** | 0,059 | 0,046 | 28,26% |
| ***Avg. Ratings/User*** | 211,02 | 171,581 | 39,439 |
| ***Avg. Ratings/Item*** | 94,469 | 74,077 | 20,392 |
| ***Max /User*** | 1743 | 1743 | 0 |
| ***Max /item*** | 1100 | 917 | 183 |
| ***MAE FunkSVD*** | 0,651 | 0,72 | -9,58% |
| ***MAE SlopeOne*** | 0,629 | 0,692 | -9,10% |
| ***RMSE FunkSVD*** | 0,826 | 0,915 | -9,73% |
| ***RMSE SlopeOne*** | 0,8 | 0,882 | -9,30% |
| ***Precision FunkSVD*** | 0,787 | 0,755 | 4,24% |
| ***Precision SlopeOne*** | 0,773 | 0,739 | 4,60% |
| ***Recall FunkSVD*** | 0,531 | 0,523 | 1,53% |
| ***Recall SlopeOne*** | 0,524 | 0,509 | 2,95% |
| ***Execution Duration*** | 713 | 632 | 12,82% |

We can see from table above that by increasing the FMV threshold, all the evaluation metrics are improved, and the sparsity of the dataset is dramatically increased. Of course, in case of a very high FMV threshold system could face the problem of not being able to select the minimum number of users required to run the recommendation algorithm.

## 5 Conclusions and future work

In this paper an extended methodology of reproducibility in recommender systems combined with source dataset optimization methods with the use or Recommender101 platform has been presented. In our previous work we have already shown that the reproducibility of results becomes achievable if the correct settings and parameters are used. This paper extends the set of parameters and achieves to increase the recommendation accuracy, to deal with the sparsity problem of big datasets and at the same time to provide the necessary tools and variables to reproduce the results of the experiments.

The initial evaluation results are promising and can assist towards this direction and our approach can be straightforwardly implemented by researchers in other libraries. Furthermore, in our future work we aim to include more custom variables in the Recommender101 and to extend the use of Fuzzy Norms in the user selection methodology.

## References

1. 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.
2. Núñez-Valdéz, E.R., et al., Implicit feedback techniques on recommender systems applied to electronic books. Computers in Human Behavior, 2012. 28(4): p. 1186-1193.
3. Porcel, C., et al., A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. Information Sciences, 2012. 184(1): p. 1-19.
4. Tan, S., et al., Using rich social media information for music recommendation via hypergraph model. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 2011. 7(1): p. 22.
5. Barragáns-Martínez, A.B., et al., A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. Information Sciences, 2010. 180(22): p. 4290-4311.
6. Costa-Montenegro, E., A.B. Barragáns-Martínez, and M. Rey-López, Which App? A recommender system of applications in markets: Implementation of the service for monitoring users' interaction. Expert systems with applications, 2012. 39(10): p. 9367-9375.
7. Bobadilla, J., F. Serradilla, and A. Hernando, Collaborative filtering adapted to recommender systems of e-learning. Knowledge-Based Systems, 2009. 22(4): p. 261-265.
8. McNally, K., et al., A case study of collaboration and reputation in social web search. ACM Transactions on Intelligent Systems and Technology (TIST), 2011. 3(1): p. 4.
9. Jalili, M., et al., Evaluating Collaborative Filtering Recommender Algorithms: A Survey. IEEE Access, 2018. 6: p. 74003-74024.
10. 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).

11. Said, A., Bellogín, A.: Comparative recommender system evaluation. Proc. 8th ACM Conf. Recomm. Syst. - RecSys ’14. 129–136 (2014).
12. 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).
13. Polatidis Nikolaos, Pimenidis Elias. Reproduction of experiments in recommender systems evaluation based on explanations. In: International Conference on Engineering Applications of Neural Networks. Springer, Cham, 2018. p. 194-200.
14. Zadeh, L. A. (1988). Fuzzy logic. Computer, 21(4), 83-93.
15. Wang, W., & Liu, X. (2011). Intuitionistic fuzzy geometric aggregation operators based on Einstein operations. International Journal of Intelligent Systems, 26(11), 1049-1075.
16. Liu, P. (2014). Some Hamacher aggregation operators based on the interval-valued intuitionistic fuzzy numbers and their application to group decision making. IEEE Transactions on Fuzzy systems, 22(1), 83-97.
17. Harper, F.M., Konstan, J.A.: The MovieLens Datasets. ACM Trans. Interact. Intell. Syst. 5, 1–19 (2015).
18. YANG, Longqi, et al. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In: Proceedings of the 12th ACM Conference on Recommender Systems. ACM, 2018. p. 279-287.
19. 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.