Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 9, No. 2, 2021, 203-212.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research Paper

Online Recommender System Considering Changes in User's Preference

Javad Hamidzadeh* and Mona Moradi

Faculty of computer engineering and information technology, Sadjad University, Mashhad, Iran.

Article Info

Article History:
Received 03 April 2020
Revised 05 June 2020
Accepted 11 December 2020

DOI:10.22044/jadm.2020.9518.2085

Keywords:
Recommender Systems, Online Learning, Natural Noise, Concept Drift.

**Corresponding author: J_Hamidzadeh@sadjad.ac.ir (J. Hamidzadeh).*

Abstract

Recommender systems extract the unseen information in order to predict the next preferences. Most of these systems use additional information such as the demographic data and previous users' ratings to predict the users' preferences but have rarely have used sequential information. In streaming recommender systems, the emergence of new patterns or disappearance of a pattern leads to inconsistencies. However, these changes are common issues due to the user's preference variations on items. Recommender systems without considering inconsistencies will suffer a poor performance. Accordingly, the present paper presents a new fuzzy rough set-based method for handling the inconsistencies in a flexible and adaptable way. The evaluations are conducted on twelve real-world datasets by the leave-one-out cross-validation method. The results of the experiments are compared with those from the other five methods, which show the superiority of the proposed method in terms of accuracy, precision, and recall.

1. Introduction

Applications of streaming data, e.g. online gaming, media streams, and IoT platforms are ever-increasing. Thus it is necessary to process such data in an online manner. Online learning usually imposes the memory and processing time restrictions. In online learning, only a single sample is introduced to a classifier at every time instant, and the patterns in the data may change over time. One of the most important problems with streaming data is *concept drift*, which causes the trained classifier to become obsolete. In order to prevent this problem, the system should be adapted to new patterns in the data. Accordingly, a classifier can be trained either incrementally by continuous updates or by retraining using the recent data. Concept drift can be categorized into five groups [1]: (1) *Abrupt drift*, which occurs when a change happens suddenly between two classification contexts; (2) *Gradual drift*, which occurs when a smooth transition emerges between two concepts; (3) *Incremental drift*, which occurs when the ratio of changes in the concepts becomes slow; (4) *Recurrent drift*, which occurs when the previously known concepts occur periodically; (5) *Blip*, which occurs when an outlier is mixed with the concept drift. Since this change is a temporary event that does not affect the future data (new features or new classes do not appear), no processing is required.

The other important problem with streaming data is the *noise*, which can be categorized into two groups: (1) *Malicious noise* that is intentionally introduced by the external agents to bias information; (2) *Natural noise* that unintentionally occurs and affects the information.

Recommender systems, as one of the applications of streaming data, are a particular type of information system that assists the users in choosing the advisable items according to their preferences [2, 3]. These systems are driven by the patterns of the users in online shopping, news websites, or click-streams in website exploring. One of the major challenges in these systems is that the data must be analyzed in a near real-time because the preferences regularly change over time, and they quickly become obsolete after some time. These inconsistencies (concept drift and natural noise) cause the recommended items to skew from the current user interest over time

and become irrelevant. Hence, the recommender system should be adapted in order to cope with the inconsistencies.

The proposed method focuses on the online sequential pattern mining for non-stationary environments. The method is incrementally updated when a new sample arrives. It aims to discover the inconsistencies presented in sequential patterns of data streams and to predict the next plausible items by considering the changes in the user's preferences. Its novelty is that it only utilizes the ratings, and does not require any additional information in order to improve the quality of the recommendations. The proposed method initially extracts knowledge from data streams using a sliding window and a three-phased strategy to discover the inconsistencies in the patterns. Once the type of inconsistency is discovered (concept drift or natural noise), a user-based collaborative filtering prediction method is employed in order to predict the correct rating. To validate the hypothesis that the corrected data improves the performance of the recommender system, experiments are conducted on twelve datasets.

The contributions of this paper can be summarized as follow:

- The proposed method focuses on the online mining sequential patterns generated in the recommender systems.
- It aims to discover and adopt the changes in user's ratings, while predicting the next plausible items by considering changes in the user's preferences.
- Once the inconsistent rating is discovered, an effective approach based on a user-based collaborative filtering method is employed in order to predict the correct rating.
- It only exploits rating knowledge, and does not depend on the additional information.
- Several experiments have been conducted to verify the performance of the proposed method. The experimental results show that the proposed method achieves better results than the other five methods for the recommendation problem.

The rest of this paper is structured as what follows. In Section 2, the related works are surveyed. The backgrounds of the sequential pattern mining and the fuzzy rough set are explained in Section 3. The proposed method is introduced in Section 4. Several experiments have been conducted on various datasets for performance comparison of the proposed method with the others. These results are discussed in Section 5. Finally, Section 6 contains conclusions and the future works.

2. Related Works

The two most used methods in recommender systems are the content-based and the collaborative filtering ones. The content-based recommender systems suggest items to a user based on the features of the items and a profile of the user's interests [3]. Collaborative filtering, as a commonly used technique in recommender systems, aims to predict the user preferences according to his/her historical information [4].

The main ideas for streaming recommender systems are often found in five categories including (1) frequent pattern mining, (2) Markov model, (3) Neural networks, (4) Matrix factorization, and (5) Supervised learning. The advantages and disadvantages of these methods are listed in Table 1.

Table 1. Streaming recommender system characteristics.

Method	Advantages	Disadvantages
Frequent pattern mining [5-13]	• Ease of use • Explainable results	• Complex configuration • Sparsity • Unscalable
Markov model [14]	• Explainable results	• Sparsity
Neural networks [16]	• Robust to sparsity • Learn long/short term dependencies	• Many parameters • Complex configuration • Limited explainable results
Matrix factorization [17]	• Robust to sparsity	• Unscalable
Supervised learning [18-28]	• Ease of use	• Feature engineering

The methods that are based on the frequent pattern mining aim to find the frequent patterns in the user's sequential actions. AprioriAll [5] and PrefixSpan [6] are the basic approaches used to find the sequential patterns. The authors in [7] have proposed a knowledge-based recommender system regarding opinion mining and rough set association rule mining to detect closed sequences. In [8], a clustering approach based on a non-overlapping sliding window for mining evolving usage data streams has been presented. It recognizes the changes in the underlying data segmentation and follow-up clusters over the subsequent time periods. Another method [9] has been presented as a recommendation model for the web personalization system by integrating web usage and web content mining. However, the system uses incremental clustering and takes into account the temporal dimension of data in the recommender systems but like any other

clustering approach, the order of the sequence is lost. In [10], an algorithm has been presented to discover the frequent itemsets through support approximation. It preserves mining accuracy well on the concept-drifting data streams. Another frequent itemset miner has been introduced in [11] that uses a sliding window technique and item weighting. In [12], a local drift degree measurement has been proposed to continuously monitor the regional density changes. Also a concept drift detector for data streams has been presented in [13] by computing the consistency of sequential error rate by utilizing the Hoeffding's Inequality.

The methods that are based on the Markov chain aim to compute the transition probabilities over the user's sequential actions, e.g. the authors in [14] have proposed a method for capturing the temporal dynamics in the dataset to adapt to a wide spectrum of collaborative filtering problems.

The recent advances in neural network-based recommender systems have gained significant attention [15]. The authors in [16] have proposed a concept drift adaptive method in order to improve the accuracy of anomaly detection on data streams by utilizing Long Short-Term Memory (LSTM).

The methods that are based on matrix factorization aim to define new inputs and loss functions over the user's sequential actions. The authors in [17] have proposed a robust to noise system that has utilized similarity upper approximation and singular value decomposition (SVD) for the generation of recommendations for the users.

The methods that are based on supervised learning aim to define the classifiers over the user's sequential actions. The authors in [18] have proposed a classifier to detect noisy ratings using a fuzzy model. In their method, noisy ratings are corrected by predicting their value with a noise-free data set. The authors in [19] have proposed a classifier to natural noise management in the group recommendation systems. In [20], a noise correction-based method has been proposed to support a recommender system in a highly sparse rating environment. In [21], a method has been proposed based on the users' interest sequences for online ranking users' ratings by extracting hidden semantics in the users' interest sequences. Patil [22] has presented a method to understand the two kinds of concept drift (gradual and abrupt) in data streams.

The authors in [23] have proposed a fuzzy model for detecting the concept drift. In [24], an adapted incremental graded multi-label classifier has been proposed for the recommendation systems by considering sparsity in ratings. In [25], a time-aware recommender system has been proposed to control which ratings are exposed to concept drift. The authors in [26] have proposed a Bayesian regression model for predicting the accuracy of each individual user feedback and also find outliers in the feedback dataset.

The authors in [27] have analyzed the customers' purchase behavior in order to extract sequential rules. They proposed a hybrid recommendation method that combined the segmentation-based sequential rule method with the segmentation-based method. The authors in [28] have proposed a tree structure for web page recommendation by exploiting web pages' semantics through websites ontology.

3. Preliminaries

This section begins with the definitions related to the sequential pattern mining and ends with the fuzzy rough set theory.

3.1. Sequential Pattern Mining

Sequential pattern mining [29] aims to find the concept drifts, frequent patterns or even abnormal patterns in a data stream. Let $I = \{i_1, i_2, ...i_m\}$ be an *itemset*, and a sequence *s* be an ordered list according to their timestamp *t*, denoted as $\langle s_1 s_2 \ldots s_n \rangle$, where each instance is an item set containing sorted items. Table 2 shows an example of an input data stream that lies on timestamps 1 to 11. This data stream composes of *User_ID*, *Timestamp*, and *Itemset*. Itemset in this table comprises of the seven items {*a,b,c,d,e,f,g*}. Table 3 presents four sequences of users. In order to mine data streams, different window models for pattern mining over data streams are used. A window *W* can be referred to as a set of all sequences between the *i*th and *j*th (where $j > i$) arrival of sequences, and the size of *W* is $|W| = j - i$. All sequences in the last *W* are utilized for data mining. Passing the time, a sequence of windows $W_{st}, \ldots, W_{prev}, W_c$ is created, where W_{st} is the window at the start of mining, W_{prev} is the window at timestamp *t*-1, and W_c is the window at the current time *t*. The aggregated window W_{Agg} is defined as all windows from the start point of mining to the window at timestamp *t-1* (W_{prev}) (see Figure 1). The count of a sequence *s* (count(*s*)) is the total number of sequences in the window containing sequence *s*. The support of a sequence *s* (*sup* (*s*)) is the portion of the total number of sequences in the

dataset that contains *s*. Table 4 indicates some sequential patterns and the number of sequences containing each pattern (called the support). A sequence is called frequent if count(*s*) ≥ *min_sup*×|*W*|, where *min_sup* is an absolute user-defined minimum support threshold (where *0* < *min_sup* < *1*), e.g. if *min_sup* is set to 0.3 and |*W*|=4, {a} and {bg} are the frequent sequences.

Table 2. An example of the input stream.

User_ID	Timestamp	Item set
1	1	{a}
2	2	{ad}
3	3	{abd}
4	4	{b}
2	5	{def}
3	6	{ac}
4	7	{abdefg}
1	8	{de}
2	9	{cb}
3	10	{efg}
4	11	{cg}

Table 3. Sequences obtained from Table 1.

User_ID	Win_ID	Sequence
1	1	{a}{de}
2	1	{ad}{def}{cb}
3	1	{abd}{ac}{efg}
4	1	{b}{cg}

Table 4. Supports of some sequences.

Sequence	Count
{a}	3
{ag}	0
{cg}	1
{bg}	2

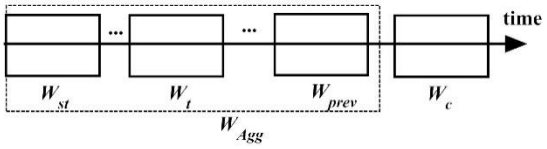

Figure. 1. Illustration of windows in the chronological order.

3.2. Fuzzy Rough Set

Fuzzy rough set [29], for each sample *s*, returns a pair of memberships that show the lower approximation membership as a degree of certainty (see Eq. (2)) and the upper approximation membership as a possibility degree of being included in target class (see Eq. (3)). The lower and upper approximation memberships are constructed by Indiscernibility Relationships (*IR*) between the samples and the amount of dependency on the target set, μ_F, respectively. *IR*, as shown in Eq. (1), measures the similarity of each pair of samples. When two samples are identical, *IR* becomes 1, and when the samples are completely different, it shows zero.

$$IR(s_i, s_j) = \underset{a \in Dimension}{\tau} (1 - |s_i(a) - s_j(a)|^2) \quad (1)$$

According to Eq. (1), the differences between the two samples in every dimension are aggregated by a *t*-norm. The result is called the indiscernibility relation between two samples under decision boundary.

$$\mu_{\underline{F_{IR,\mu_F}}}(s_i) = \inf_{s_j \in T, s_i \neq s_j} I(IR(s_i, s_j), \mu_F(s_j)) \quad (2)$$

$$\mu_{\overline{F_{IR,\mu_F}}}(s_i) = \sup_{s_j \in T, s_i \neq s_j} \tau(IR(s_i, s_j), \mu_F(s_j)) \quad (3)$$

where *T* denotes the dataset. As shown in the above equations, fuzzy operators, Implicator (*I*) and *t*-norm (τ) combine two basic elements and result in several outputs. Then "*inf*" and "*sup*" select one of these outcomes as the final result. Hence, the outliers and noise data can change the lower and upper approximation memberships in a wide range. Authors in [31] have proposed the adjusted versions of the memberships using Order Weigh Average (OWA) instead of "*inf*" and "*sup*". These forms of memberships are presented as follow:

$$\mu_{\underline{F_{IR,\mu_F}}}(s_i) = OWA_{\min}\, I(IR(s_i, s_j), \mu_F(s_j)) \quad (4)$$

$$\mu_{\overline{F_{IR,\mu_F}}}(s_i) = OWA_{\max}\, \tau(IR(s_i, s_j), \mu_F(s_j)) \quad (5)$$

4. Methodology

The proposed method is introduced in sub-section 4.1. Also the time complexity is explained in sub-section 4.2.

4.1. Proposed Method

The proposed method focuses on the online mining sequential patterns generated in recommender systems. It aims to discover and adopt the changes in user's ratings, while predicting the next plausible items by employing a proper adaptive way. The steps of the proposed method are presented in Steps 1.

Steps 1. Steps of the proposed method.

Step 1:	Construct the current window, W_c Create a cache table, named *freq_cur*, to store the frequent sequences of W_c Construct the aggregated window, W_{Agg} Create a cache table, named *freq_hist*, to store frequent sequences of W_{Agg} Create a cache table, named *cache_pattern*, to store frequent sequences of both *freq_cur* and *freq_hist*
Step 2:	Find frequent sequences by employing the method presented in [32]
Step 3:	Sliding window update: Update W_c and *freq_cur* Update W_{Agg} and *freq_hist* Update *cache_pattern*
Step 4:	Compute the dissimilarity between W_{Agg} and W_c using Eq. (8)
Step 5:	Determine the type of changes using Fig. 2
Step 6:	Predict the next preferences using Eq. (13)

Now the discovering process is introduced. Given a pre-defined value for *min_sup*, the frequent sequences are identified within each window

using the method introduced in [32]. For each mining round, the frequent sequences of the current window (window at timestamp *t*) W_c, and the aggregated window, W_{Agg}, are stored in *freq_cur* and *freq_hist*, respectively. Besides, *freq_cur* and *freq_hist* are merged to form a cache table called *cache_pattern*. The proposed method finds the changes (i.e. noise and concept drift) by checking the dissimilarity between the frequent sequences in W_{Agg} and W_c. Once the window slides, the support value of sequences is updated. In order to discover the changes, the support of frequent sequences within the current windows $\left(sup_{W_c}\right)$ and all the previously observed windows $\left(sup_{W_{Agg}}\right)$ are computed as follow:

$$sup_{W_c} = \sum_{i=timestamp(t)}^{timestamp(t)+|W_c|} sup_i \,, \tag{6}$$

$$sup_{W_{Agg}} = \sum_{i=timestamp(st)}^{timestamp(st)+|W_{Agg}|} sup_i \,, \tag{7}$$

The dissimilarity between sup_{W_c} and $sup_{W_{Agg}}$ is computed by Eq. (8) as follows.

$$Dis\_sim = \frac{\arccos\left(\dfrac{sup_{W_c}.sup_{W_{Agg}}}{\left|sup_{W_c}\right|\left|sup_{W_{Agg}}\right|}\right)}{\pi}. \tag{8}$$

Since the magnitude of a vector cannot be a negative number, *Dis_sim* belongs to the interval [0, 0.05].

Then the proposed method requires to determine the type of changes (natural noise and concept drift). The state diagram of the proposed method is shown in Figure. 2, including three main phases: (1) *Steady phase*, (2) *Alarm phase*, and (3) *Detection phase*. More specifically, these are explained:

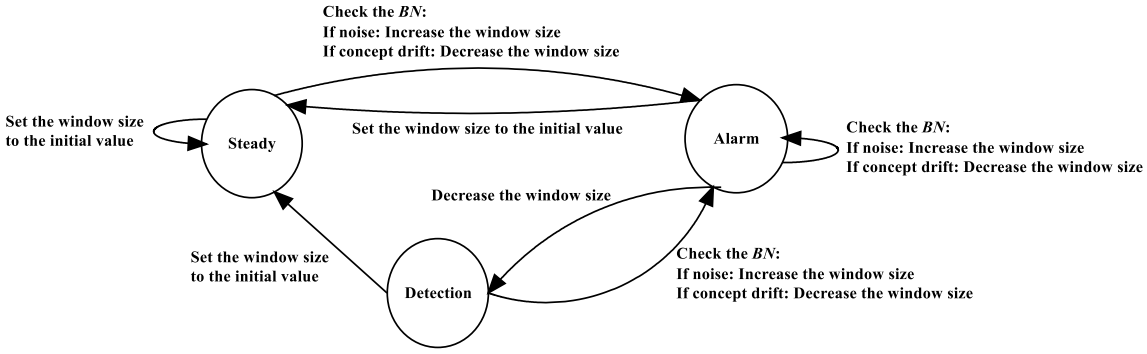

Figure. 2. State diagram of the proposed method.

1. ***Steady phase*:** This phase occurs when *Dis_sim* is lower than threshold σ_1. σ_1 is a pre-defined value that displays the maximum acceptable dissimilarity to remain in the steady phase. During this phase, the window size does not change, and it is equal to the window size at the starting point of mining.
2. ***Alarm phase*:** This phase occurs when *Dis_sim* is between two thresholds, σ_1 and σ_2. σ_2 is a pre-defined value that determines the minimum dissimilarity for detecting concept drift. During this phase, data reveal slight changes. In order to quantify the user's inconsistency degree (natural noise and concept drift), the proposed method utilizes the fuzzy rough set approach. Rough set theories have revealed themselves as one of the most successful approaches to deal with imprecise, inconsistent, and incomplete information, and also to identify the temporal patterns. Besides, the fuzzy set theory can be effective to consider flexibility in the learning process. Considering this fact, the boundary region *BN* as a feasible decision is defined by Eq. (9).

$$\begin{aligned} BN &= \mu_{\overline{F_{IR,\mu F}}}(p_i) - \mu_{\underline{F_{IR,\mu F}}}(p_i) \\ &= \underset{p_j \in T, p_i \neq p_j}{OWA_{\max}} \tau(IR(p_i, p_j), \mu_F(p_j)) \\ &\quad - \underset{p_j \in T, p_i \neq p_j}{OWA_{\min}} \tau(IR(p_i, p_j), \mu_F(p_j)) \end{aligned} \tag{9}$$

where p is a frequent sequence presented in the dataset T. If the size of the boundary region is less than or equal to the adjustable parameter α, $0 \le \alpha \le 1$, it reveals that this uncertainty is due to the natural noise; otherwise, the concept drift is suspected. The selected optimal α varies under different criteria. For making strict decisions, the greater α (closer to 1) should be chosen. In this case, the changes are considered as noise. Also for making soft decisions, the lower α (closer to 0) should be chosen. In this case, the changes are considered as concept drift. It is important to note that in the alarm phase, the proposed method predicts the change in the running mining round and modifies the size of

windows depending on the type of the change (natural noise or concept drift). In the case that noise is detected, the adverse effect of noise should be mitigated. Hence, the flexible window size $|W|$ will be increased in order to provide a significant training set to achieve good generalization and reliable results. In this case, the window size must be modified as follows:

$$|W_c| = |W_{prev}| + \lambda \times |W_{prev}| \quad (10)$$

where $|W_c|$ is the size of the current window and λ $(0 < \lambda \leq 1)$ is a pre-defined extending factor. On the other hand, in the case that the concept drift is suspected, the flexible window size $|W|$ will be decreased as follows:

$$|W_c| = \rho \times |W_{prev}| \quad (11)$$

where ρ, $0 < \rho < 1$ is a pre-defined forgetting factor to shrink the window. The reason behind shrinking the window size is that, passing the time and arriving new data, the old data must be ignored due to (1) expiration of the old data and (2) limitation in the storage resource. Accordingly, all the stored frequent sequences in the *cache_pattern* are removed. Also, the starting point for frequent pattern mining is set to the current timestamp.

*3.**Detection phase:*** If Dis_sim is greater than threshold σ_2, the concept drift is detected. In this case, training should be done by recent instances. The window size $|W|$ will be decreased using Eq. (11). Also due to the changes in data streams, the starting point for frequent pattern mining is set to the current timestamp, and all the stored frequent sequences in cache_pattern are removed. Procedure 1 provides a brief description to distinguish the type of changes.

As shown in Figure. 2, there is no transition from the steady phase to the detection phase. Since the window slides by moving one window forward at each time, the swift reaction to concept drift does not happen.

Now, regarding the changes in the user's preferences, the next plausible items should be predicted. Hence, a user-based collaborative filtering method that relies on the user-to-user similarity measure is used.

Regarding timestamp *t*, *User_ID*, and the current window, W_c, the user similarity (see Eq. (12)) and user's rating of an item (see Eq. (13)) are calculated.

$$sim\_rating(u,y) = \frac{\sum_{i \in I_{u,y}} \left(R^t_{u,i} - \overline{R}^t_u\right).\left(R^t_{y,i} - \overline{R}^t_y\right)}{\sqrt{\sum_{i \in I_{u,y}} \left(R^t_{u,i} - \overline{R}^t_u\right)^2} . \sqrt{\sum_{i \in I_{u,y}} \left(R^t_{y,i} - \overline{R}^t_y\right)^2}} \quad (12)$$

where $R^t_{u,i}$ is the rating of user *u* to item *i* at timestamp *t*, and $\overline{R}^t_u$ is the averaged ratings of user *u* at timestamp *t*. Next, the rank of user *u* to item *i* is predicted as follows:

$$P_{u,i} = \overline{R}^t_u + \frac{\sum_{y \in N_u} \left(R^t_{y,i} - \overline{R}^t_y\right) * sim\_rating(u,y)}{\sqrt{\sum_{y \in N_u} |sim\_rating(u,y)|}} \quad (13)$$

where $R^t_{y,i}$ is the rating of neighbor *y* to item *i* at timestamp *t*, and $\overline{R}^t_y$ is the averaged ratings of neighbor *y* at timestamp *t*. Finally, regarding the ratings predicted for item *i*, the recommendations are made for user *u*.

Procedure 1: Pseudo-code of change detection.

Initialization:

window size $|W|$, thresholds σ_1, σ_2, parameter α, extending factor λ, forgetting factor ρ.

For each timestamp t:

1. Start with computing *Dis_sim* using Eq. (8) for each moving window.
2. The sequence can be found in three different conditions:
   - 2.1. Steady phase: If $Dis\_sim < \sigma_1$, the window size is kept fix and sets to its initial value. The window continues its movement on the data sequence to find frequent sequences using the proposed method in [32].
   - 2.2. Alarm phase: If $\sigma_1 \leq Dis\_sim \leq \sigma_2$, compute *BN* (using Eq. (9)).
     - 2.2.1. If $BN \leq \alpha$: natural noise is detected. Increase the window (using Eq. (10)).
     - 2.2.2. Else, the occurrence of concept drift is suspected. In this case, (a) remove all the frequent sequences stored in *cache_pattern*, (b) set the starting point for mining to the current timestamp, (c) shrink the window (using Eq. (11)).
   - 2.3. Detection phase: When $Dis\_sim > \sigma_2$, concept drift is detected. In this case, (a) remove all the frequent sequences stored in *cache_pattern*, (b) set the starting point for mining to the current timestamp, (c) shrink the window (using Eq.(11)).
3. Slide the window on the data stream.

4.2. Time Complexity

The time complexity of the proposed method is presented in Table 5. Accordingly, it is computed as a function of the input size *n*, the number of features *k*, the depth of the decision tree *d*, the number of fuzzy sets on *i*th input F_i, and the cardinality of the region including the data points *R*.

Table 5. Time complexity.

Step	Computations
Finding frequent sequences	$O(nkd)$
Calculating dissimilarities	$O(n)$
Discovering the type of changes	$O(nRF_i)$
Total	$O(nkd + nRF_i)$

5. Experimental Results

In this section, the performance of the proposed method is evaluated on twelve real-world datasets. The detailed information of the collected datasets is listed in sub-section 5.1. The experimental

setup is covered in sub-section 5.2. Extensive experiments are carried out in order to evaluate the proposed method against five other methods. These results obtained and their interpretations are described in sub-section 5.3.

5.1. Datasets

The experiments are performed on twelve datasets adapted for change detection in the distribution of data. The characteristics of the datasets used are presented in Table 6.

Table 6. Dataset characteristics.

Dataset	# sequence	# item	Address
Cardiac Arrhythmia	452	279	https://archive.ics.uci.edu
Mushrooms	8,124	22	https://archive.ics.uci.edu
Spam	9,324	40,000	https://archive.ics.uci.edu
CTI	13,745	683	http://dampa.cdm.depaul.edu
FIFA	20,450	2,990	http://france98.com
BMSWebView1 (Gazelle)	59,601	497	http://kdd.org
SEA Concepts	60,000	3	http://liaad.up.pt
MovieLens 100K	100,000	1,682	https://grouplens.org
MovieLens 1M	1,000,000	4,000	https://grouplens.org
Jester	4,100,000	100	https://grouplens.org/datasets/jester/
Ciao	278,483	99,746	http://dvd.ciao.co.uk
MSNBC	989,818	17	http://archive.ics.uci.edu

5.2. Experimental Setup

The proposed method can be tuned dynamically during the learning process. The window size w at initialization is set to 5, and the thresholds σ_1, σ_2 are set to 0.1 and 0.4, respectively. For all datasets, the numerical sequences are considered as the features. As mentioned earlier, these sequences are the frequently occurring patterns that are obtained using the method presented in [31]. The adjustable parameter α is set to 0.5. The extending factor λ and the forgetting factor ρ are set to 20%. Also the four criteria accuracy (see Eq. (14)), precision (see Eq. (15)), recall (see Eq. (16)), and computational time are used as the evaluation metrics.

$$accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{14}$$

$$precision = \frac{t_p}{t_p + f_p} \tag{15}$$

$$recall = \frac{t_p}{t_p + f_n} \tag{16}$$

where t_p is the true positive (relevant item is recommended), f_p is the false positive (irrelevant item is recommended), f_n is the false negative (relevant item is not recommended), and t_n is the true negative (irrelevant item is not recommended). Besides, the computational time is regarded as the averaged time required in order to perform a computational process on an Intel processor at 2.30 GHz with 4 GB of RAM.

5.3. Results and Interpretation

The experimental results are expressed in Tables 7-14. These tables indicate the classification performance of the proposed method and five other methods [13, 14, 17, 21, 22] in terms of the accuracy, precision, recall, and computational time. The quality of each method is evaluated by the leave-one-out cross-validation method. In this case, only one data point is picked as a test set. Then the method is performed on all the remaining, complementary instances, and evaluation is carried out.

Table 7. Accuracy (%).

Dataset	[14]	[22]	[17]	[21]	[13]	Proposed method
Cardiac Arrhythmia	56.01 (5)	58.04 (4)	64.53 (2)	51.61 (6)	57.19 (3)	**66.89 (1)**
Mushroom	66.67 (6)	71.43 (4)	71.51 (3)	71.14 (5)	72.11 (2)	**76.19 (1)**
Spam	57.54 (3)	48.57 (6)	57.24 (4)	54.29 (5)	58.23 (2)	**68.57 (1)**
CTI	71.35 (5)	71.28 (6)	71.43 (3)	**76.19 (1)**	73.45 (2)	71.42 (4)
FIFA	55.57 (3)	54.64 (5)	55.26 (4)	51.85 (6)	59.31 (2)	**63.98 (1)**
BMSWebView1 (Gazelle)	51.95 (4)	49.35 (5)	53.25 (3)	35.06 (6)	63.76 (2)	**68.89 (1)**
SEA Concepts	64.58 (5)	62.50 (5)	**72.92 (1)**	54.17 (6)	67.92 (3)	69.77 (2)
MovieLens 100K	58.78 (3)	60.81 (4)	67.57 (2)	58.11 (6)	**68.02 (1)**	65.89 (3)
MovieLens 1M	71.92 (3)	64.76 (4)	62.91 (5)	59.23 (6)	72.11 (2)	**80.67 (1)**
Jester	73.53 (2)	61.65 (4)	59.83 (6)	60.54 (5)	70.74 (3)	**82.33 (1)**
Ciao	56.95 (4)	55.85 (5)	54.25 (6)	57.60 (3)	68.14 (2)	**70.68 (1)**
MSNBC	59.69 (4)	59.41 (5)	64.72 (3)	54.60 (6)	67.85 (2)	**79.82 (1)**
Average rank	3.92	4.75	3.5	5.08	2.17	**1.5**

Table 8. Results of HSD test for accuracy.

Method	Average accuracy	Significance from
[14]	62.04	Proposed method
[22]	59.86	Proposed method
[17]	62.95	Proposed method
[21]	57.03	[13], Proposed method
[13]	66.57	[21]
Proposed method	**72.08**	[14], [22], [17], [21], [13]

Moreover, since the average rank provides a reasonable comparison of the methods, the performance of each method is ranked. The best result is highlighted in **bold**. Furthermore, ANalysis Of Variance (ANOVA) (specifically, Turkey's Honestly Significance Difference (HSD)) test is employed in order to show the meaningfulness of the experiments. ANOVA promises rejection of experiments' meaningfulness as the null hypothesis, which supposes that there is no noticeable difference between pair of methods.

One method is accepted as a distinguished one if the null hypothesis becomes refused at a 5% significance level.

Table 9. Precision (%).

Dataset	[14]	[22]	[17]	[21]	[13]	Proposed method
Cardiac Arrhythmia	51.72 (3)	52.17 (4)	**61.90 (1)**	41.97 (6)	51.27 (5)	60.15 (2)
Mushroom	50.48 (6)	55.56 (1)	54.55 (3)	54.48 (4)	53.21 (5)	**66.87 (1)**
Spam	52.38 (4)	45.62 (6)	52.63 (3)	49.91 (5)	71.01 (2)	**72.83 (1)**
CTI	50.78 (5)	51.86 (2)	50.35 (6)	**57.00 (1)**	50.84 (4)	51.79 (3)
FIFA	53.85 (3)	52.09 (5)	53.40 (4)	49.77 (6)	56.93 (2)	**61.64 (1)**
BMSWebView1 (Gazelle)	50.06 (4)	46.67 (5)	51.12 (3)	31.84 (6)	56.02 (2)	**62.75 (1)**
SEA Concepts	53.85 (4)	51.85 (5)	**61.54 (1)**	44.32 (6)	54.30 (3)	56.92 (2)
MovieLens 100K	51.79 (5)	53.85 (4)	60.87 (2)	50.22 (6)	**61.38 (1)**	58.83 (3)
MovieLens 1M	51.29 (6)	52.86 (3)	52.35 (4)	51.45 (5)	54.73 (2)	**61.93 (1)**
Jester	53.73 (2)	50.32 (6)	53.48 (3)	52.04 (5)	53.34 (4)	**60.84 (1)**
Ciao	51.79 (3)	50.35 (5)	48.73 (6)	51.61 (4)	**71.36 (1)**	64.55 (2)
MSNBC	50.94 (4)	50.65 (5)	55.21 (3)	45.72 (6)	70.17 (2)	**72.84 (1)**
Average rank	4.08	4.25	3.25	5	2.75	**1.58**

Table 10. Results of HSD test for precision.

Method	Average precision	Significance from
[14]	51.89	[13], Proposed method
[22]	51.15	[13], Proposed method
[17]	54.68	Proposed method
[21]	48.36	[13], Proposed method
[13]	58.71	[14], [22], [21], Proposed method
Proposed method	**62.66**	[14], [22], [17], [21]

Table 11. Recall (%).

Dataset name	[14]	[22]	[17]	[21]	[13]	Proposed method
Cardiac Arrhythmia	52.76 (2)	41.13 (5)	42.13 (3)	29.63 (6)	42.01 (4)	**66.67 (1)**
Mushroom	57.32 (6)	65.97 (4)	70.36 (2)	**71.43 (1)**	69.65 (3)	61.14 (5)
Spam	**66.37 (1)**	54.32 (4)	57.80 (2)	56.25 (3)	48.92 (6)	50.04 (5)
CTI	65.18 (3)	64.90 (4)	62.28 (5)	65.34 (2)	60.28 (6)	**66.67 (1)**
FIFA	51.34 (3)	50.65 (4)	47.17 (5)	46.15 (6)	60.77 (2)	**61.54 (1)**
BMSWebView1 (Gazelle)	38.18 (3)	31.06 (6)	36.14 (4)	32.43 (5)	83.74 (2)	**86.49 (1)**
SEA Concepts	71.17 (2)	70.33 (3)	**74.65 (1)**	57.89 (6)	67.83 (5)	69.82 (4)
MovieLens 100K	43.72 (6)	51.63 (5)	**62.85 (1)**	52.38 (4)	61.66 (2)	59.73 (3)
MovieLens 1M	63.87 (5)	70.54 (3)	71.54 (2)	57.99 (6)	68.75 (4)	72.47 (1)
Jester	65.39 (5)	71.26 (2)	70.22 (3)	58.04 (6)	68.03 (4)	74.02 (1)
Ciao	45.94 (5)	54.21 (3)	49.08 (6)	58.54 (2)	49.85 (4)	**77.72 (1)**
MSNBC	58.98 (4)	61.81 (3)	70.85 (2)	56.82 (5)	48.56 (6)	**82.93 (1)**
Average rank	3.75	3.83	3	4.33	4	**2.08**

The results of the accuracy comparison of the methods are presented in Table 7. As shown below, the proposed method has the average rank 1. Although the proposed method gains good results for most datasets, it achieves rank 2 in SEA Concepts and MovieLens 100K. Besides, it takes rank 4 for the CTI data set. The results of HSD test (see Table 8) illustrate the superiority of the proposed method in comparison with the competitors.

The precision results are presented in Table 9. As shown below, the proposed method has the average rank 1. Table 10 proves that the proposed method is significantly better than all the competitors except [13]. The results of the recall comparison are listed in Table 11. As shown below, the proposed method has the best average rank. Table 12 reveals that there is a significant difference between the proposed method and [21] in terms of recall metric.As shown in Table 13, the computational time has also been reported to assess the dynamics of the proposed method. As the average rank shows, the proposed method has the second-lowest computational time. As shown in Table 14, it is significantly better than [14].

Table 12. Results of HSD test for recall.

Method	Average recall	Significance from
[14]	56.68	-
[22]	57.32	-
[17]	59.59	-
[21]	53.57	Proposed method
[13]	60.84	-
Proposed method	69.10	[21]

Table 13. Computational time (s).

Dataset	[14]	[22]	[17]	[21]	[13]	Proposed method
Cardiac Arrhythmia	88.23 (6)	55.35 (3)	**37.24 (1)**	81.24 (5)	66.03 (4)	51.20 (2)
Mushroom	96.23 (6)	59.03 (3)	61.34 (4)	89.13 (5)	57.19 (2)	**42.23 (1)**
Spam	83.40 (5)	66.12 (4)	**31.54 (1)**	51.23 (3)	89.23 (6)	48.23 (2)
CTI	89.34 (6)	51.76 (4)	32.65 (2)	**32.10 (1)**	85.78 (5)	45.54 (3)
FIFA	85.23 (6)	48.32 (2)	**28.34 (1)**	76.12 (4)	83.35 (5)	51.34 (3)
BMSWebView1 (Gazelle)	90.35 (6)	59.35 (3)	**30.24 (1)**	81.34 (5)	64.11 (4)	52.53 (2)
SEA Concepts	56.75 (6)	36.11 (5)	24.76 (2)	26.23 (4)	25.20 (3)	**21.43 (1)**
MovieLens 100K	81.23 (6)	60.21 (4)	43.01 (2)	73.21 (5)	47.44 (3)	**36.12 (1)**
MovieLens 1M	92.51 (6)	71.55 (3)	66.01 (2)	86.39 (5)	**65.04 (1)**	73.79 (4)
Jester	97.72 (2)	**83.58 (1)**	99.66 (3)	109.17 (4)	149.94 (6)	109.88 (5)
Ciao	68.23 (6)	48.23 (5)	31.23 (2)	35.23 (3)	48.14 (4)	**26.34 (1)**
MSNBC	62.35 (6)	40.34 (5)	29.46 (2)	30.23 (3)	32.09 (4)	**28.53 (1)**
Average rank	5.58	3.5	**1.92**	3.92	3.92	2

Table 14. Results of HSD test for computational time.

Method	Average computational time	Significance from
[14]	82.63	[17], Proposed method
[22]	56.66	-
[17]	**42.96**	[14]
[21]	64.30	-
[13]	67.80	-
Proposed method	48.93	[14]

6. Conclusions and Future Works

Streaming recommender systems aim to predict the next item from the sequential users' online behavior. The sequential data can be changed over time; thus there is a pressing need for the methods that adapt to these inconsistencies including natural noise and concept drift. The present paper designs a novel method for tracking the changes and predicting the next preferences dynamically in the streaming recommender systems. It discovers the frequent patterns in the sequences; then these patterns are utilized in order to obtain the user-based similarities, and further, to predict the best next item. A good tool to deal with the inconsistencies could be the fuzzy rough set theory because they are natively adapted to imprecise, inconsistent, incomplete information, which usually undermines the performance of the recommender systems. Regarding the fuzzy rough set theory, a three-phased strategy is designed for discovering the changes in sequential patterns. The proposed method is based on two main ideas: (1) finding frequent sequences, and (2) utilizing a classification task. The former leads to explainable sequences and removes the uninformative data. However, it causes limited scalability and data sparsity. Even though the latter is easy to implement, its performance depends on the performance of the classifier and the quality of the selected features. The experiments on twelve datasets demonstrate the effectiveness of the proposed method in the streaming recommender system area, in which it outperforms in terms of accuracy, precision, and recall. Also the performance of the computational time is acceptable.

In the future works, the authors tend to improve the computational time and utilize new data resources such as demographic data in order to reduce the adverse effect of cold start problems. Moreover, the optimized concept drift adaptive method is required to be investigated to further improve the detection performance in this paper.

References

[1] B. Krawczyk, and A. Cano, "Online ensemble learning with abstaining classifiers for drifting and noisy data streams," *Applied Soft Computing,* vol. 68, pp. 677-692, 2018.

[2] M. Moradi, and J. Hamidzadeh, "Ensemble-based Top-k Recommender System Considering Incomplete Data," *Journal of AI and Data Mining,* vol. 7, no. 3, pp. 393-402, 2019.

[3] F. Amato, V. Moscato, A. Picariello *et al.*, "SOS: a multimedia recommender system for online social networks," *Future generation computer systems,* vol. 93, pp. 914-923, 2019.

[4] V. Maihami, D. Zandi, and K. Naderi, "Proposing a novel method for improving the performance of collaborative filtering systems regarding the priority of similar users," *Physica A: Statistical Mechanics and its Applications,* vol. 536, pp. 121021, 2019.

[5] R. Srikant, and R. Agrawal, "Mining sequential patterns: Generalizations and performance

improvements," *Advances in Database Technology—EDBT'96*, pp. 1-17, 1996.

[6] J. Pei, J. Han, B. Mortazavi-Asl *et al.*, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 11, pp. 1424-1440, 2004.

[7] H. Zang, Y. Xu, and Y. Li, "Non-redundant sequential association rule mining and application in recommender systems." pp. 292-295.

[8] A. Da Silva, Y. Lechevallier, and F. A. de Carvalho, "CAMEUD: clustering approach for mining evolving usage data." p. 3.

[9] C. Rana, and S. Jain, "A recommendation model for handling dynamics in user profile," *Distributed Computing and Internet Technology*, pp. 231-241, 2012.

[10] C.-W. Li, and K.-F. Jea, "An approach of support approximation to discover frequent patterns from concept-drifting data streams based on concept learning," *Knowledge and information systems*, vol. 40, no. 3, pp. 639-671, 2014.

[11] G. Lee, U. Yun, and K. H. Ryu, "Sliding window based weighted maximal frequent pattern mining over data streams," *Expert Systems with Applications*, vol. 41, no. 2, pp. 694-708, 2014.

[12] A. Liu, Y. Song, G. Zhang *et al.*, "Regional concept drift detection and density synchronized drift adaptation."

[13] M. M. W. Yan, "Accurate detecting concept drift in evolving data streams," *ICT Express*, 2020.

[14] R. Zhang, and Y. Mao, "Movie Recommendation via Markovian Factorization of Matrix Processes," *IEEE Access*, 2019.

[15] S. Zhang, L. Yao, A. Sun *et al.*, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1-38, 2019.

[16] R. Xu, Y. Cheng, Z. Liu *et al.*, "Improved Long Short-Term Memory based anomaly detection with concept drift adaptive method for supporting IoT services," *Future Generation Computer Systems*, 2020.

[17] R. Mishra, P. Kumar, and B. Bhasker, "A web recommendation system considering sequential information," *Decision Support Systems*, vol. 75, pp. 1-10, 2015.

[18] R. Yera, J. Castro, and L. Martínez, "A fuzzy model for managing natural noise in recommender systems," *Applied Soft Computing*, vol. 40, pp. 187-198, 2016.

[19] J. Castro, R. Yera, and L. Martínez, "An empirical study of natural noise management in group recommendation systems," *Decision Support Systems*, vol. 94, pp. 1-11, 2017.

[20] S. Bag, S. Kumar, A. Awasthi *et al.*, "A noise correction-based approach to support a recommender system in a highly sparse rating environment," *Decision Support Systems*, vol. 118, pp. 46-57, 2019.

[21] W. Cheng, G. Yin, Y. Dong *et al.*, "Collaborative Filtering Recommendation on Users' Interest Sequences," *PloS one*, vol. 11, no. 5, pp. e0155739, 2016.

[22] M. M. Patil, "Handling Concept Drift in Data Streams by Using Drift Detection Methods," *Data Management, Analytics and Innovation*, pp. 155-166: Springer, 2019.

[23] Q. Zhang, D. Wu, G. Zhang *et al.*, "Fuzzy user-interest drift detection based recommender systems." pp. 1274-1281.

[24] K. Laghmari, C. Marsala, and M. Ramdani, "An adapted incremental graded multi-label classification model for recommendation systems," *Progress in Artificial Intelligence*, vol. 7, no. 1, pp. 15-29, 2018.

[25] A. Alzogbi, "Time-aware Collaborative Topic Regression: Towards Higher Relevance in Textual Item Recommendation." pp. 10-23.

[26] A. Kangasrääsiö, Y. Chen, D. Głowacka *et al.*, "Interactive modeling of concept drift and errors in relevance feedback." pp. 185-193.

[27] D.-R. Liu, C.-H. Lai, and W.-J. Lee, "A hybrid of sequential rules and collaborative filtering for product recommendation," *Information Sciences*, vol. 179, no. 20, pp. 3505-3519, 2009.

[28] T. T. S. Nguyen, H. Y. Lu, and J. Lu, "Web-page recommendation based on web usage and domain knowledge," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2574-2587, 2014.

[29] R. Agrawal, and R. Srikant, "Mining sequential patterns." pp. 3-14.

[30] D. Dubois, and H. Prade, "Rough fuzzy sets and fuzzy rough sets," *International Journal of General System*, vol. 17, no. 2-3, pp. 191-209, 1990.

[31] N. Verbiest, C. Cornelis, and F. Herrera, "FRPS: A fuzzy rough prototype selection method," *Pattern Recognition*, vol. 46, no. 10, pp. 2770-2782, 2013.

[32] E. Loekito, J. Bailey, and J. Pei, "A binary decision diagram based approach for mining frequent subsequences," *Knowledge and Information Systems*, vol. 24, no. 2, pp. 235-268, 2010.

سیستم پیشنهاددهنده برخط با درنظرگرفتن تغییرات در سلیقه‌ی کاربر

جواد حمیدزاد* و منا مرادی

دانشکده مهندسی کامپیوتر و فن آوری اطلاعات، دانشگاه سجاد، مشهد، خراسان رضوی، ایران.

ارسال ۲۰۲۰/۰۴/۰۳؛ بازنگری ۲۰۲۰/۰۶/۰۵؛ پذیرش ۲۰۲۰/۱۲/۱۱

چکیده:

سیستم‌های پیشنهاددهنده برای پیش‌بینی ترجیحات بعدی کاربران اطلاعات مشاهده‌نشده را استخراج می‌کنند. بیشتر این سیستم‌ها از اطلاعات اضافی مانند داده‌های جمعیتی و امتیازات پیشین کاربران برای پیش‌بینی ترجیحات و سلایق استفاده نموده و به ندرت از داده‌های متوالی استفاده می‌کنند. در سیستم‌های پیشنهاددهنده مبتنی بر جریان داده، پدیدار شدن الگوهای جدید یا ناپدید شدن یک الگو منجر به ناسازگاری شده هرچند وجود چنین ناسازگاری‌هایی به دلیل تغییرات سلایق کاربر امری عادی تلقی می‌شود. درنظر نگرفتن این ناسازگاری‌ها منجر به عملکرد ضعیف سیستم می‌شود. از این‌رو، مقاله حاضر به ارائه‌ی یک روش جدید مبتنی بر مجموعه تقریب فازی به منظور مدیریت این ناسازگاری‌ها به شیوه‌ای انعطاف‌پذیر می‌پردازد. ارزیابی‌ها بر روی دوازده مجموعه داده دنیای واقعی با روش اعتبارسنجی یک طرفه انجام شده است. مقایسه‌ی نتایج آزمایش‌ها با پنج روش دیگر، نشان‌دهنده برتری روش پیشنهادی از نظر صحت، دقت، یادآوری و زمان محاسباتی است.

کلمات کلیدی: سیستم‌های پیشنهاددهنده، یادگیری برخط، نویز طبیعی، رانش مفهوم.