

RAPS: A Recommender Algorithm Based on Pattern Structures

Dmitry I. Ignatov¹ and Denis Kornilov¹

National Research University Higher School of Economics
dignatov@hse.ru
<http://www.hse.ru>

Abstract. We propose a new algorithm for recommender systems with numeric ratings which is based on Pattern Structures (RAPS). As the input the algorithm takes rating matrix, e.g., such that it contains movies rated by users. For a target user, the algorithm returns a rated list of items (movies) based on its previous ratings and ratings of other users. We compare the results of the proposed algorithm in terms of precision and recall measures with Slope One, one of the state-of-the-art item-based algorithms, on Movie Lens dataset and RAPS demonstrates the best or comparable quality.

Keywords: Formal Concept Analysis, Pattern Structures, Recommender Systems, Collaborative Filtering, RAPS, Slope One

1 Introduction and related work

Formal Concept Analysis (FCA)[1] is a powerful algebraic framework for knowledge representation and processing [2,3]. However, in its original formulation it deals with mainly Boolean data. Even though original numeric data can be represented by so called multi-valued context, it requires concept scaling to be transformed to a plain context (i.e. a binary object-attribute table). There are several extensions of FCA to numeric setting like Fuzzy Formal Concept Analysis [4,5]. In this paper, to recommend particular user items of interest we use Pattern Structures, an extension of FCA to deal with data that have ordered descriptions. In fact, we use interval pattern structures that were proposed in [6] and successfully applied, e.g., in gene expression data analysis [7].

The task of recommending items to users according to their preferences expressed by ratings of previously used items became extremely popular during the last decade partially because of famous NetFlix 1M\$ competition [8]. Numerous algorithms were proposed to this end. In this paper we will mainly study item-based approaches. Our main goal is to see whether FCA-based approaches are directly applicable to the setting of recommender systems with numeric data. Previous approaches used concept lattices for navigation through the recommender space and allowed to recommend relevant items faster than online computation in user-based approach, however it requires expensive offline computations and a substantial storage space [9]. Another approach tries to effectively use Boolean factorisation based on formal concepts and follows user-based k-nearest neighbours strategy [10]. A parameter-free approach that exploits a neighbourhood of the object concept for a particular user also proved its effectiveness

[11] but it has a predecessor based on object-attribute biclusters [12] that also capture the neighbourhood of every user and item pair in an input formal context. However, it seems that within FCA framework item-based techniques for data with ratings have not been proposed so far. So, the paper bridges the gap.

The paper is organised as follows. In Section 2, basic FCA definitions and interval pattern structures are introduced. Section 3 describes SlopeOne [13] and RAPS with examples. In Section 4, we provide the results of experiments with time performance and precision-recall evaluation for MovieLens dataset. Section 5 concludes the paper.

2 Basic definitions

Formal Concept Analysis. First, we recall several basic notions of Formal Concept Analysis (FCA) [1]. Let G and M be sets, called the set of objects and attributes, respectively, and let I be a relation $I \subseteq G \times M$: for $g \in G$, $m \in M$, gIm holds iff the object g has the attribute m . The triple $\mathbb{K} = (G, M, I)$ is called a (*formal*) *context*. If $A \subseteq G$, $B \subseteq M$ are arbitrary subsets, then the *Galois connection* is given by the following *derivation operators*:

$$\begin{aligned} A' &= \{m \in M \mid gIm \text{ for all } g \in A\}, \\ B' &= \{g \in G \mid gIm \text{ for all } m \in B\}. \end{aligned} \quad (1)$$

The pair (A, B) , where $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$ is called a (*formal*) *concept* (of the context K) with *extent* A and *intent* B (in this case we have also $A'' = A$ and $B'' = B$).

The concepts, ordered by $(A_1, B_1) \geq (A_2, B_2) \iff A_1 \supseteq A_2$ form a complete lattice, called *the concept lattice* $\mathfrak{B}(G, M, I)$.

Pattern Structures. Let G be a set of objects and D be a set of all possible object descriptions. Let \sqcap be a similarity operator. It helps to work with objects that have non-binary attributes like in traditional FCA setting, but those that have complex descriptions like intervals or graphs. Then (D, \sqcap) is a meet-semi-lattice of object descriptions. Mapping $\delta : G \rightarrow D$ assigns an object g the description $d \in (D, \sqcap)$.

A triple $(G, (D, \sqcap), \delta)$ is a pattern structure. Two operators $(\cdot)^\square$ define Galois connection between $(2^G, \subseteq)$ and (D, \sqcap) :

$$A^\square = \bigcap_{g \in A} \delta(g) \text{ for } A \subseteq G \quad (2)$$

$$\begin{aligned} d^\square &= \{g \in G \mid d \sqsubseteq \delta(g)\} \text{ for } d \in (D, \sqcap), \text{ where} \\ d &\sqsubseteq \delta(g) \iff d \sqcap \delta(g) = d. \end{aligned} \quad (3)$$

For a set of objects A operator 2 returns the common description (pattern) of all objects from A . For a description d operator 3 returns the set of all objects that contain d .

A pair (A, d) such that $A \subseteq G$ and $d \in (D, \sqcap)$ is called a pattern concept of the pattern structure $(G, (D, \sqcap), \delta)$ iff $A^\square = d$ and $d^\square = A$. In this case A is called a pattern extent and d is called a pattern intent of a pattern concept (A, d) . Pattern concepts are partially ordered by $(A_1, d_1) \leq (A_2, d_2) \iff A_1 \subseteq A_2 (\iff d_2 \sqsubseteq d_1)$. The set of all pattern concepts forms a complete lattice called a pattern concept lattice.

Intervals as patterns. It is obvious that similarity operator on intervals should fulfill the following condition: two intervals should belong to an interval that contains them. Let this new interval be minimal one that contains two original intervals. Let $[a_1, b_1]$ and $[a_2, b_2]$ be two intervals such that $a_1, b_1, a_2, b_2 \in \mathbb{R}$, $a_1 \leq b_1$ and $a_2 \leq b_2$, then their similarity is defined as follows:

$$[a_1, b_1] \sqcap [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)].$$

Therefore

$$\begin{aligned} [a_1, b_1] \sqsubseteq [a_2, b_2] &\iff [a_1, b_1] \sqcap [a_2, b_2] = [a_1, b_1] \\ &\iff [\min(a_1, a_2), \max(b_1, b_2)] = [a_1, b_1] \\ &\iff a_1 \leq a_2 \text{ and } b_1 \geq b_2 \iff [a_1, b_1] \supseteq [a_2, b_2] \end{aligned}$$

Note that $a \in \mathbb{R}$ can be represented by $[a, a]$.

Interval vectors as patterns. Let us call p-adic vectors of intervals as interval vectors. In this case for two interval vectors of the same dimension $e = \langle [a_i, b_i] \rangle_{i \in [1, p]}$ and $f = \langle [c_i, d_i] \rangle_{i \in [1, p]}$ we define similarity operation via the intersection of the corresponding components of interval vectors, i.e.:

$$e \sqcap f = \langle [a_i, b_i] \rangle_{i \in [1, p]} \sqcap \langle [c_i, d_i] \rangle_{i \in [1, p]} \iff e \sqcap f = \langle [a_i, b_i] \sqcap [c_i, d_i] \rangle_{i \in [1, p]}$$

Note that interval vectors are also partially ordered:

$$e \sqsubseteq f \iff \langle [a_i, b_i] \rangle_{i \in [1, p]} \sqsubseteq \langle [c_i, d_i] \rangle_{i \in [1, p]} \iff [a_i, b_i] \sqsubseteq [c_i, d_i]$$

for all $i \in [1, p]$.

3 Recommender Algorithms

3.1 Slope One

Slope One is one of the common approaches to recommendations based on collaborative filtering. However, it demonstrates comparable quality with more complex and resource demanding algorithms [13]. As it was shown in [14], SlopeOne has the highest recall on MovieLens and Netflix datasets and acceptable level of precision: “Overall, the algorithms that present the best results with these metrics are SVD techniques, tendencies-based and slope one (although its precision is not outstanding).”

We use this algorithm for comparison purposes.

Slope One deals with rating matrices as input data. In what follows the data contains movies ratings by different users. That is $M = \{m_1, m_2, \dots, m_n\}$ is a set of movies, $U = \{u_1, u_2, \dots, u_k\}$ is a set of users. The rating matrix can be represented by many-valued formal context (U, M, R, I) , where $R = \{1, 2, 3, 4, 5, *\}$ is a set of possible ratings and a triple $(u, m, r) \in I$ means that the user u marked by the rating r the movie m . Whenever it is suitable we also use r_{ij} notation for rating of movie m_j by user u_i .

In case a user u has not rated a movie m , we use $m(u) = r = *$, i.e. missing rating.

Let us describe the algorithm step by step.

1. The algorithm takes a many valued context of all users' ratings, the target user u_t for which it generates recommendations. It also requires *left_border* and *right_border* for acceptable level of ratings, i.e. if one wants to receive all movies with ratings between 4 and 5, then left and right borders should be 4 and 5 respectively. The last pair of parameters: one needs to set up minimal and maximal scores (*min_border* and *max_border*) that are acceptable for our data. It means that if the algorithm predicts rating 6.54 as a score and maximal score is bounded by 5, then 6.54 should be treated as 5.
 2. The algorithm finds the set of all movies evaluated by the target user $S(u_t)$.
 3. For every non-evaluated movie $m_j \in M \setminus S(u_t)$ by u_t execute step 4), and by so doing calculate the predicted rating for the movie m_j . After that go to step 5).
 4. For every evaluated movie $m_i \in S(u_t)$ by u_t calculate $S_{j,i}(U \setminus \{u_t\})$, the set of users that watched and evaluated movies m_i and m_j . In case $S_{j,i}(U \setminus \{u_t\})$ is non-empty, that is $|S_{j,i}(U \setminus \{u_t\})| > 0$, calculate the deviation: $dev_{j,i} = \sum_{u_k \in S_{j,i}(U \setminus \{u_t\})} \frac{r_{k,j} - r_{k,i}}{|S_{j,i}(U \setminus \{u_t\})|}$ and add i to R_j .
- After all current deviations found, calculate the predicted rating: $P(u_t)_j = \frac{1}{|R_j|} \sum_{i \in R_j} (dev_{j,i} + r_{t,i})$, where $R_j = \{i | m_i \in S(u_t), i \neq j, |S_{j,i}(U \setminus \{u_t\})| > 0\}$. In case R_j is empty, the algorithm cannot make a prediction.
5. By this step Slope One found all predicted ratings $P(u_t)$ for movies from $M \setminus S(u_t)$. The algorithm recommends all movies with predicted ratings in the preferred range $left_border \leq P(u_t)_j \leq right_border$, taking into account minimal and maximal allowed values.

If one needs top- N ranked items, she can sort the predicted scores from the resulting set in decreasing order and select first N corresponding movies.

Example 1.

Consider execution of Slope One on the dataset from Table 1.

Table 1. Example of data for Slope One

user \ movie	m_1	m_2	m_3
u_1	5	3	2
u_2	3	4	*
u_3	*	2	5

Let us try to predict the rating for u_3 and movie m_1 .

1. Let $left_border = 4$, $right_border = 5$, $min_border = 1$, and $max_border = 5$.
2. We find $S(u_3) = \{m_2, m_3\}$, the set of evaluated movies by the target user.
3. $M \setminus S(u_3) = \{m_1\}$
4. $S_{1,2}(U \setminus \{u_3\}) = \{u_1, u_2\}$
 $dev_{1,2} = \frac{(r_{1,1}-r_{1,2})+(r_{2,1}-r_{2,2})}{(|\{u_1, u_2\}|)} = ((5-3) + (3-4))/2 = 0.5$
 $S_{1,3}(U \setminus \{u_3\}) = \{u_1\}$
 $dev_{1,3} = (r_{1,1} - r_{1,3})/(|\{u_1\}|) = (5-2)/1 = 3$
 $R_1 = \{2, 3\}$
 $P(u_3)_1 = 1/|R_j|(dev_{1,2} + r_{3,2} + dev_{1,3} + r_{3,3}) = 1/2(0.5 + 2 + 3 + 5) = 5.25$
5. Taking into account the maximal rating boundary, the algorithm predicts 5 for movie m_1 , and therefore recommends user u_3 to watch it.

3.2 RAPS

Our approach, RAPS (Recommender Algorithm based on Pattern Structures), works with the same many valued context as Slope One.

Let us describe the algorithm.

1. It takes the context (U, M, R, I) with all ratings, and a target user u_t . It also requires $left_border$ and $right_border$ for preferred ratings, i.e. if one wants to get all movies rated in range from 4 to 5, then $left_border = 4$ and $right_border = 5$.
2. Define the set of movies $M_t = \{m_{t_1}, \dots, m_{t_q}\}$ that the target user u_t liked, i.e. the ones that she evaluated in the range $[left_border, right_border]$.
3. For each movie $m_{t_i} \in M_t$ apply eq. 3. and find the set of users that liked the movie $A_{t_i} = [left_border, right_border]_{m_{t_i}}^\square$ for $1 \leq i \leq q$. As a result one has the set of user subsets: $\{A_{t_1}, \dots, A_{t_q}\}$.
4. For each A_{t_i} , $1 \leq i \leq q$ apply eq. 2 to find its description; in our case it is a vector of intervals $d_{t_i} = A_{t_i}^\square = \langle [a_1^{t_i}, b_1^{t_i}], \dots, [a_n^{t_i}, b_n^{t_i}] \rangle$ for $1 \leq i \leq q$. Note that, in case a particular user u_x from A_{t_i} has not rated m_y , i.e. $r_{x,y} = *$, then the algorithm does not take it into account.
5. At the last step compute the vector $\mathbf{r} = (R_1, \dots, R_n) \in \mathbb{N}^n$ (or \mathbb{R}^n in general case), where

$$R_j = |\{i | 1 \leq i \leq q, [a_j^{t_i}, b_j^{t_i}] \subseteq [left_border, right_border]\}|, \text{ i.e.}$$

for each movie m_j the algorithm counts how many of its descriptions $[a_j^{t_i}, b_j^{t_i}]$ are in $[left_border, right_border]$. If $R_j > 0$, then the algorithm recommends watching the movie.

Top- N movies with the highest ratings can be selected in similar way.

Let us shortly discuss the time computational complexity. Step 2 requires $O(|M|)$ operations, steps 3, 4 and 5 perform within $O(|M||U|)$ each. Therefore, the algorithm time complexity is bounded by $O(|M||U|)$.

Example 2

Consider execution of RAPS on the tiny dataset from Table 2.

Let us find a recommendation for user u_7 .

Table 2. Example of data for RAPS

user\movie	m_1	m_2	m_3	m_4	m_5	m_6
u_1	5	3	1	3	5	3
u_2	4	4	1	5	4	3
u_3	5	*	*	3	*	4
u_4	*	3	4	*	2	4
u_5	4	*	4	5	4	*
u_6	3	4	5	5	*	3
u_7	5	4	2	*	*	*

1. The input of the algorithm: $t = 7$, $left_border = 4$ and $right_border = 5$.
2. $M_7 = \{m_1, m_2\}$
3. $A_1 = [4, 5]_{m_1}^\square = \{u_1, u_2, u_3, u_5\}$
 $A_2 = [4, 5]_{m_2}^\square = \{u_2, u_6\}$
- 4.

$$d_1 = A_1^\square = \langle [a_1^1, b_1^1], [a_2^1, b_2^1], [a_3^1, b_3^1], [a_4^1, b_4^1], [a_5^1, b_5^1], [a_6^1, b_6^1] \rangle = \\ = \langle [4, 5], [3, 4], [1, 4], [3, 5], [4, 5], [3, 4] \rangle$$

For example, interval $[a_6^1, b_6^1]$ is found as follows:

$$[a_6^1, b_6^1] = [\min(r_{1,6}, r_{2,6}, r_{3,6}, r_{5,6}), \max(r_{1,6}, r_{2,6}, r_{3,6}, r_{5,6})] = \\ = [\min(3, 3, 4, *), \max(3, 3, 4, *)] = [\min(3, 3, 4), \max(3, 3, 4)] = [3, 4].$$

The rest intervals are found in similar way.

$$d_2 = A_2^\square = \langle [3, 4], [4, 4], [1, 5], [5, 5], [4, 4], [3, 3] \rangle$$

5. Taking into account the left and right bounds, the algorithm recommends movies m_1 and m_5 from d_1 and m_2, m_4 and m_5 from d_2 . Therefore $R = (1, 1, 0, 1, 2, 0)$, i.e. without already assessed movies by u_7 , we recommend her to watch m_4 and m_5 .

4 Experimental evaluation

4.1 Data

For our experimentation we have used freely available data from MovieLens website¹. The data collection was gathered within The GroupLens Research Project of Minnesota University in 1997–1998. The data contains 100000 ratings for 1682 movies by 943 different users. Each user rated no less than 20 movies. That is we have 100000 tuples in the form:

user id | item id | rating | timestamp.

Each tuple shows user id, movie id, the rating she gave to the movie and time when it happened.

¹ <http://grouplens.org/datasets/movielens/>

4.2 Quality assessment

Firstly, for quality assessment of Slope One and RAPS we used precision and recall measures. Note that we cannot use Mean Absolute Error (MAE) directly, since RAPS actually assume a whole interval like [4,5] for a particular movie, not a number. We select 20% of users to form our test set and for each test user we split her rated movies into two parts: the visible set and the hidden set. The first set consists of 80% rated movies, and the second one contains the remaining 20%. Moreover, to make the comparison more realistic, movies from the first set were evaluated earlier than those from the second one. It means that first we sort all ratings of a given user by timestamp and then perform splitting.

There is a general testing scheme based on bimodal cross-validation from [15], which seems to us the most natural and realistic: users from the test set keep only $x\%$ of their rated movies, and the remaining $y\%$ of their ratings are hidden. Thus, by considering each test user in this way, we model a real user whose ratings to other movies are not yet clear, but at the same time we have all ratings' information about the training set of users. In other words, we hide only rectangle of size $x\%$ of test users by $y\%$ of hidden items. One can vary x and y during the investigation of the behaviour of methods under comparison, where the size of top-N recommended list is set to be equal to $y\%$. The part of hidden items can be selected randomly or by timestamp (preferably for realistic scenario). Note that there is no a gold standard approach to test recommender systems, however, there are validated sophisticated schemes [16]. The main reason is the following: with only off-line data in hands we cannot verify whether the user will like a not yet seen movie irrespective of assumption that she has seen our recommendation. However, for real systems there is a remedy such as A/B testing, which is applicable only in online setting [17].

The adjusted precision and recall are defined below:

$$precision = \frac{|\{relevant\ movies\} \cap \{retrieved\ movies\} \cap \{test\ movies\}|}{|\{retrieved\ movies\} \cap \{test\ movies\}|} \quad (4)$$

$$recall = \frac{|\{relevant\ movies\} \cap \{retrieved\ movies\} \cap \{test\ movies\}|}{|\{relevant\ movies\} \cap \{test\ movies\}|} \quad (5)$$

These measures allow us to avoid the uncertainty since we do not know how actually a particular user would assess a recommended movie. However, in real recommender system, we would rather ask a user whether the recommendation was relevant, but in our off-line quality assessment scheme we cannot do that. In other words, we assume that for a test user at the moment of assessment there are no movies except the training and test ones.

Another issue, which is often omitted in papers on recommender systems, is how to avoid uncertainty when denominators in Precision and Recall are equal to zero (not necessarily simultaneously).

To define the measures precisely based on the peculiarities of recommendation task and common sense, we use two types of the definitions for cases when the sets of retrieved and relevant movies for particular user and recommender are empty.

Precision and Recall of the first type are defined as follows:

- If the sets of relevant movies and retrieved ones are empty, then $Precision = 0$ and $Recall = 1$.
- If the set of relevant movies is empty, but the set of retrieved ones is not, then $Precision = 0$ and $Recall = 1$.
- If we have the non-empty set of relevant movies, but the set of retrieved movies is empty, then $Precision = 0$ and $Recall = 0$.

Precision of the second type is less tough, but Recall remains the same:

- If the sets of relevant movies and retrieved ones are empty, then $Precision = 1$ (since we should not recommend any movie and the recommender has not recommended anything).
- If the set of relevant movies is empty, but the set of retrieved ones is not, then $Precision = 0$.
- If we have the non-empty set of relevant movies, but the set of retrieved movies is empty, then $Precision = 1$ (since the recommender has not recommended anything, its output does not contain any non-relevant movie).

4.3 Results

We have performed three series of tests:

1. A movie is worth to watch if its predicted mark is 5 (i.e. it is [5,5]).
2. A mark is good if it is from [4,5].
3. Any mark from [3,5] is good.

All the tests are performed in OS X 10.9.3 with Intel Core i7 2.3 GHz and 8 Gb of memory. The algorithm were implemented in MATLAB – R2013a. The results are presented in Table 3. Note that the reported Precision and Recall are of the first type.

Table 3. RAPS vs Slope One Results

Algorithm name	Preference Interval	Average time, s	Average precision	Average recall	F1-measure
RAPS	[5,5]	3.62	19.42	50.52	28.06
Slope One	[5,5]	18.90	1.57	23.41	2.94
RAPS	[4,5]	18.23	55.61	63.33	59.22
Slope One	[4,5]	18.90	53.99	30.39	38.89
RAPS	[3,5]	32.98	80.11	83.65	81.84
Slope One	[3,5]	18.90	83.81	81.88	82.83

The criteria are average execution time in seconds, average precision and recall. From the table one can see RAPS is drastically better than Slope One by the whole set of criteria in [5,5]. For [4,5] interval both approaches have comparable time and precision, but Slope One has two times lower recall. For [3,5] interval the algorithms demonstrate similar values of precision and recall but RAPS 1.5 times slower on average.

However, since the compared approaches are different from the output point view (RAPS provides the user with an interval of possible ratings but SlopeOne does it by a single real number), we perform thorough comparison varying the lower bound of acceptable recommendations and using both types of the adjusted precision and recall measures.

From Fig. 1 one can conclude that RAPS dominates SlopeOne in most cases by Recall. As for Precision, even though for $[5,5]$ interval RAPS is significantly better, after lower bound of 4.4 SlopeOne shows comparable but slightly better Precision in most cases.

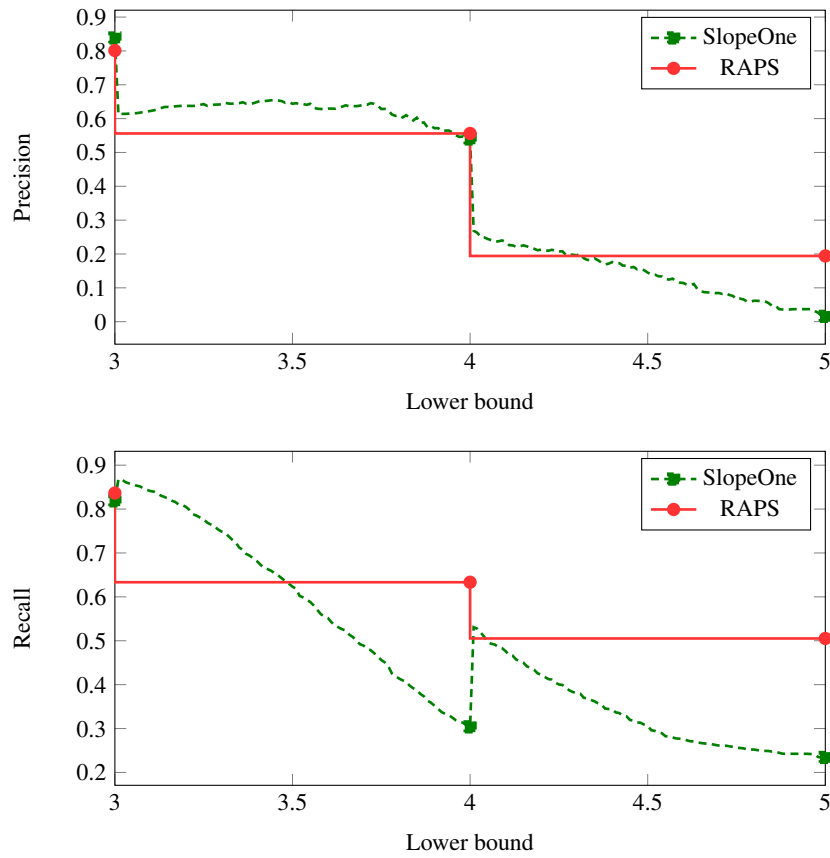


Fig. 1. Precision and Recall of the first type for RAPS and SlopeOne for the varying lower bound

From Fig. 2 one can see that SlopeOne is significantly better in terms of precision. Only on the interval $[3,5]$ the difference between SlopeOne and RAPS is negligible (the lower bound value equals 3). The reasonable explanations is as follows: for

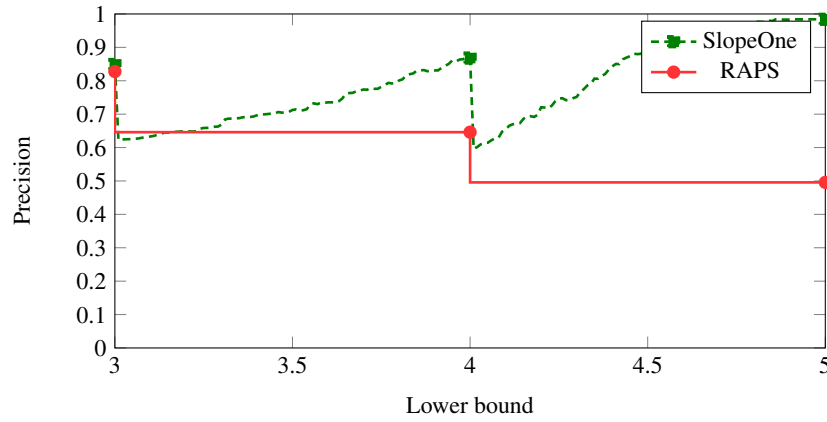


Fig. 2. Precision of the second type for RAPS and SlopeOne for the varying lower bound

SlopeOne there are more cases when $\{\text{retrieved movies}\} = \emptyset$ irrespective of the size of $\{\text{relevant movies}\}$. Remember that in such cases Precision of the second type is equal to 1. In other words SlopeOne is really more precise (or even concise): in such cases it just does not recommend anything. However, it can be hardly judged in movie recommendation domain that a recommender is good when it does not recommend.

We can conclude that the proposed recommender technique based on pattern structures has its right to be used. Since the Slope One algorithm was exploited in real recommender systems [13], we can suggest our technique for usage as well.

5 Conclusion and further work

In this paper we proposed the technique for movie recommendation based on Pattern Structures (RAPS). Even though this algorithm is oriented to movie recommendations, it can be easily used in other recommender domains where users evaluate items.

The performed experiments (RAPS vs Slope One) showed that recommender system based on Pattern Structures demonstrates acceptable precision, better recall in most cases and reasonable execution time.

Of course, in future RAPS should be compared with other recommender techniques to make a final conclusion about its applicability. An interplay between interval-based recommendations and regression-like ones deserves a more detailed treatment as well.

The further work can be continued in the following directions:

1. Further modification and adjustment of RAPS.
2. Development of the second variant of Pattern Structures based recommender. There is a conjecture that for the second derivation operation (operator Galois from eq.3) being applied to more than one movie with high marks we may obtain relevant predictions as well.

3. Comparison with existing popular techniques, e.g. SVD and SVD++.

Acknowledgments We would like to thank Mehdi Kaytoue, Sergei Kuznetsov and Sergei Nikolenko for their comments, remarks and explicit and implicit help during the paper preparations. The first author has been supported by the Russian Foundation for Basic Research grants no. 13-07-00504 and 14-01-93960 and made a contribution within the project “Data mining based on applied ontologies and lattices of closed descriptions” supported by the Basic Research Program of the National Research University Higher School of Economics. We also deeply thank the reviewers for their comments and remarks that helped.

References

1. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. 1st edn. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1999)
2. Poelmans, J., Ignatov, D.I., Kuznetsov, S.O., Dedene, G.: Formal concept analysis in knowledge processing: A survey on applications. *Expert Syst. Appl.* **40**(16) (2013) 6538–6560
3. Poelmans, J., Kuznetsov, S.O., Ignatov, D.I., Dedene, G.: Formal concept analysis in knowledge processing: A survey on models and techniques. *Expert Syst. Appl.* **40**(16) (2013) 6601–6623
4. Belohlávek, R.: What is a fuzzy concept lattice? II. In Kuznetsov, S.O., Slezak, D., Hepting, D.H., Mirkin, B., eds.: Rough Sets, Fuzzy Sets, Data Mining and Granular Computing - 13th International Conference, RSFDGrC 2011, Moscow, Russia, June 25-27, 2011. Proceedings. Volume 6743 of Lecture Notes in Computer Science., Springer (2011) 19–26
5. Poelmans, J., Ignatov, D.I., Kuznetsov, S.O., Dedene, G.: Fuzzy and rough formal concept analysis: a survey. *Int. J. General Systems* **43**(2) (2014) 105–134
6. Ganter, B., Kuznetsov, S.: Pattern structures and their projections. In Delugach, H., Stumme, G., eds.: Conceptual Structures: Broadening the Base. Volume 2120 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2001) 129–142
7. Kaytoue, M., Kuznetsov, S.O., Napoli, A., Duplessis, S.: Mining gene expression data with pattern structures in formal concept analysis. *Information Sciences* **181**(10) (2011) 1989 – 2001 Special Issue on Information Engineering Applications Based on Lattices.
8. Bell, R.M., Koren, Y.: Lessons from the netflix prize challenge. *SIGKDD Explorations* **9**(2) (2007) 75–79
9. du Boucher-Ryan, P., Bridge, D.: Collaborative recommending using formal concept analysis. *Knowledge-Based Systems* **19**(5) (2006) 309 – 315
10. Ignatov, D.I., Nenova, E., Konstantinova, N., Konstantinov, A.V.: Boolean matrix factorisation for collaborative filtering: An fca-based approach. In Agre, G., Hitzler, P., Krisnadhi, A.A., Kuznetsov, S.O., eds.: Artificial Intelligence: Methodology, Systems, and Applications - 16th International Conference, AIMSA 2014, Varna, Bulgaria, September 11-13, 2014. Proceedings. Volume 8722 of Lecture Notes in Computer Science., Springer (2014) 47–58
11. Alqadah, F., Reddy, C., Hu, J., Alqadah, H.: Biclustering neighborhood-based collaborative filtering method for top-n recommender systems. *Knowledge and Information Systems* (2014) 1–17
12. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J.: Concept-based biclustering for internet advertisement. In Vreeken, J., Ling, C., Zaki, M.J., Siebes, A., Yu, J.X., Goethals, B., Webb, G.I., Wu, X., eds.: 12th IEEE International Conference on Data Mining Workshops, ICDM Workshops, Brussels, Belgium, December 10, 2012, IEEE Computer Society (2012) 123–130

13. Lemire, D., Maclachlan, A.: Slope one predictors for online rating-based collaborative filtering. In: Proceedings of the 2005 SIAM International Conference on Data Mining, 471–475
14. Cacheda, F., Carneiro, V., Fernández, D., Formoso, V.: Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Trans. Web* **5**(1) (February 2011) 2:1–2:33
15. Ignatov, D.I., Poelmans, J., Dedene, G., Viaene, S.: A new cross-validation technique to evaluate quality of recommender systems. In Kundu, M.K., Mitra, S., Mazumdar, D., Pal, S.K., eds.: *Perception and Machine Intelligence - First Indo-Japan Conference, PerMIn 2012*, Kolkata, India, January 12-13, 2012. Proceedings. Volume 7143 of *Lecture Notes in Computer Science.*, Springer (2012) 195–202
16. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems. RecSys '10, New York, NY, USA, ACM (2010) 39–46
17. Radlinski, F., Hofmann, K.: Practical online retrieval evaluation. In Serdyukov, P., Braslavski, P., Kuznetsov, S.O., Kamps, J., Rüger, S.M., Agichtein, E., Segalovich, I., Yilmaz, E., eds.: *Advances in Information Retrieval - 35th European Conference on IR Research, ECIR 2013*, Moscow, Russia, March 24-27, 2013. Proceedings. Volume 7814 of *Lecture Notes in Computer Science.*, Springer (2013) 878–881