UDC 519.876

**RUDENSKYY R., RUDENSKA V.**

# PROBABILISTIC APPROACHES TO BIG DATA PROCESSING IN COLLABORATIVE FILTERING BASED RECOMMENDATION SYSTEMS

This article investigates the state and prospective of recommendation systems implementation under the conditions of Big Data development. The article reveals key problems that arise when collaborative filtering based recommendations are implemented and suggests approach to overcome these problems. The proposed approach is based on ideas of expanding original user-item rating matrix and implementing minhash trick to estimate Jaccard similarity measure. Ability to track users' behavior and account for it while decision making revealed the new field related to market research, data and computer science that turned into online recommendations system.

**Keywords:** recommendation systems, big data, collaborative filtering, Jaccard similarity, minhash.

**Formulation of the problem**. Huge volumes and velocity of information needed to be processed for good decisions making turned computers to become efficient and evidently the only reasonable solution for this problem. With growing social networks and net communications that led to expansion of agents and available information fostered the big data to arise on scene and become the mainstream of data processing for the latest several decades. Tons of information stored and available online on the one hand and big enough processing capacities on the other hand made it possible to turn advertisement to be addressed not to abstract but to fairly concrete and well known customer. Ability to track users' behavior and account for it while decision making revealed the new field related to market research, data and computer science that turned into online recommendations system.

**Analysis of recent publications and research.** In the early 1990s one of the most prominent technique for dealing with personal recommendations based on analysis of the behavior of large number of people became the one named collaborative filtering. Automated collaborative filtering systems soon followed, automatically locating relevant opinions and aggregating them to provide recommendations [1].

For example GroupLens [2] used this technique to identify Usenet articles which are likely to be interesting to a particular user. Users only needed to provide ratings or perform other observable actions. One of the most widely-known application of recommender system technologies is Amazon.com. Based on purchase history, browsing history, and the item a user is currently viewing, they recommend items for the user to consider purchasing. Thus recommender systems and collaborative filtering became a top topic of human–computer interaction and machine learning researchers.

The recent spike of activity in researches of recommender algorithms was motivated by Netflix in 2006 when they announced the $1 M prize to improve the state of movie recommendation. The objective of this competition was to build a recommender algorithm that could beat their internal CineMatch algorithm [3].

**The purpose of the article.** Before stating the problem let's consider the main concepts that lie behind recommendations based on collaborative filtering.

Collaborative filtering techniques depend on the triple (User, Item, Rating). This triple means that users express preferences for different items. A preference expressed by a user for an item is called a rating. These ratings can be of different forms, for example like/dislike, differently scaled integer or real numbers.

**The main material research.** The set of all rating triples forms a sparse matrix referred to as the ratings matrix. The matrix has n rows, where n is the number of Users and m columns, where m is the number of Items. Each element of the matrix is either a certain rate that reflects a user's reaction on an item or nothing in case when a user has no opinion of an item (table 1).

ISSN 2221-8440

ЧАСОПИС ЕКОНОМІЧНИХ РЕФОРМ № 1(25)/2017

Table 1

**User-item rating matrix form**

| | | Items | | | | |
|---|---|---|---|---|---|---|
| | | Item 1 | Item 2 | Item 3 | | Item m |
| Users | User 1 | r11 | - | r13 | | r1m |
| | User 2 | - | | r23 | | r2m |
| | | | | | | |
| | User n | rn1 | rn2 | rn3 | | - |

*Source:* developed by authors

The fundamental assumption behind collaborative filtering (CF) is that if some user is agree about the opinion of other users on some set of items, then he or she will likely agree about other items that has not been rated yet.

User-user CF is a straightforward algorithmic interpretation of the core conceptual assumption of collaborative filtering: find other users whose past rating behavior is similar to that of the current user and use their ratings on other items to predict what the current user will like.

Besides the rating matrix, a user–user CF system requires a similarity function:

$$s: U \times U \to R,$$

To generate recommendations for a user $u_i$ CF predicts ratings for items not yet rated by user. Predicted rating is usually computed as weighted average of ratings where similarities are used as the weights.

$$\pi_{ij} = \frac{\sum_{k \neq i} R_{kj} w_{ik}}{\sum_{k \neq i} w_{ik}}, \qquad (1)$$

where $\pi_{ij}$ - is the predicted rating of item $j$ for user $i$;

$R_{kj}$ - ratings of item $j$ by user $k$;

$w_{ik}$ - similarity between user $i$ and $k$.

Once predictions have been computed, the recommendation is formed as top N items that have the highest predicted rating.

It is worth mentioning that formula (1) in fact has mostly theoretical meaning while to be used in actual recommending engines it is the subject to different manipulations. The most common manipulations are subtracting the user's mean rating that compensates for the situation when some users tend to give constantly higher or lower ratings than others; normalize user ratings to z-scores by dividing the offset from mean rating by the standard deviation of each user's ratings, thereby compensating for both users differing in rating spread and mean rating [4].

Another point is how to estimate user-user similarity function. One pretty straightforward approach is to use simple correlation between rows of User-item rating matrix (see table 1). But as matrix becomes more sparse (and it always does when number of items grows substantially), the less informative correlation is and it becomes a real problem when we have millions of items whilst average number of rated items by a certain user is measured by dozens or even less.

Another problem we face with user-user CF is that user-user similarity matrix has extremely large dimensions and size as well. Say we have 10M users, which is not that much for modern online trading systems. In this case the size of user-user similarity matrix can be estimated as 10M times 10M times 8 (as we store 8 bytes in each cell) divided by 2 as it is enough to have only upper diagonal values) that yields over 363 Pb.

The size problem can be somehow solved by switching to sparse matrices and in case of quite sparse data it really helps but the problem with computing complexity that is $O(n^2)$ still persists.

For the purposes of recommendations making and solve problems mentioned above in this article we propose to use Jaccard similarity measure. With respect to recommendation making users similarity can be estimated as follows:

$$J\left(u_i, u_j\right) = \frac{\left|u_i \cap u_j\right|}{\left|u_i \cup u_j\right|}. \qquad (2)$$

Here $\left|u_i \cap u_j\right|$ stands for number of items equally rated by two users;

$\left|u_i \cup u_j\right|$ - total number or items rated by at least one of the users.

In case of binary ratings (likes/dislikes) Jaccard similarity measure behaves like Cosine similarity that well fits our goal even for quite sparse data. A bit different situation is observed when the rates have certain scale, say 1-5 or 1-9. In this case Jaccard similarity measure behaves quite aggressively in the sense that it does not account for the rates difference unlike for example correlation and distinguish exactly two states: rates are equal or rates are not equal.

As a workaround of this problem we propose to supplement each rate provided by user with certain range of rates. For example if we have some item rated with 3 and 4 by two different users original Jaccard similarity measure would consider them as completely different and add zero to the numerator in formula (2).

Substitution of original rates with respective ranges allows adjusting Jaccard similarity measure to be less aggressive. For example, if we have five point scale and introduce the range of width three we get the following results (table 2).

So, as one can see from table 2 if users' rates differ by one point the proposed approach (for given 5 point scale and range of length 3) generates similarity of ½, if rates' difference is 2 – the similarity is 1/5 and if difference is 3 or more we get zero similarity exactly like if we'd been using the original Jaccard similarity formula.

One can easily see that for given range of length $(Rg)$ and ranks' difference $(\Delta_r)$ the similarity of two different points $(p_1, p_2)$ can be calculated as follows:

$$sim(p_1, p_2) = \frac{\max(Rg - \Delta_r(p_1, p_2), 0)}{Rg + \Delta_r(p_1, p_2)}, \quad (3)$$

Table 2

**Adjusted Jaccard similarity measure**

| Scale | User 1 rate | User 2 rate | User 3 rate | User 1 Range | User 2 Range | User 3 Range |
|---|---|---|---|---|---|---|
| 1 | | | | X | | |
| 2 | X | | | X | X | |
| 3 | | X | | X | X | X |
| 4 | | | X | | X | X |
| 5 | | | | | | X |
| Jaccard Similarity | | | | | | |
| | Original | | | Adjusted | | |
| | User 1 | User 2 | User 3 | User 1 | User 2 | User 3 |
| User 1 | 1 | 0 | 0 | 1 | 1/2 | 1/5 |
| User 2 | 0 | 1 | 0 | 1/2 | 1 | 1/2 |
| User 3 | 0 | 0 | 1 | 1/5 | 1/2 | 1 |

*Source:* developed by authors

whereas original Jaccard similarity approach considers the similarity of points $(p_1, p_2)$ like:

$$sim\left(p_1, p_2\right) = \begin{cases} 1, p_1 = p_2 \\ 0, p_1 \neq p_2 \end{cases} \qquad (4)$$

So, this approach seems to solve the problem of estimating similarities on sparse data even for non- binary ratings. Another problem that has been posted is the dimensionality problem. To solve this one we propose to apply probabilistic algorithms.

It is worth mentioning that one can find a variety of methods from the machine learning and artificial intelligence literature devoted to dimensionality reduction like clustering, principal component analysis, singular value

ЧАСОПИС ЕКОНОМІЧНИХ РЕФОРМ № 1(25)/2017 ISSN 2221-8440

ЧАСОПИС ЕКОНОМІЧНИХ РЕФОРМ №1(25)/2017 ISSN 2221-8440

decomposition [5, 6] but they are out of scope of this article.

As we led similarity measure to Jaccard similarity it is reasonable to implement probabilistic approach for Jaccard similarity estimation known as MinHash technique.

MinHash (or the min-wise independent permutations locality sensitive hashing scheme) was invented by Andrei Broder (1997), [7] and initially used in the AltaVista search engine to detect duplicate web pages and eliminate them from search results [8].

For our purposes instead of distinct documents we have rows of user-item ratings matrix where original ratings are supplemented with corresponding ranges of ratings (see table 2 as an example).

Let $U_1$ and $U_2$ are two different rows from the adjusted user-ratings matrix and $h$ is the hash function that maps each member of $U_1$ and $U_2$ into integers.

Let $h_{\min}(U_1)$ is the minimal value we get when $(h)$ is applied to each member of $U_1$ and $h_{\min}(U_2)$ is the minimal value we get when $(h)$ is applied to each member of $U_2$. It can be shown that (for details see [9]):

$$J(U_1,U_2)=\Pr\left(h_{\min}(U_1)=h_{\min}(U_2)\right),$$

where $\Pr(\cdot)$ means probability of the event $(\cdot)$.

To estimate this probability we have two options one is to use random permutations from $U_1$ and $U_2$ and calculate the frequency when two minhash values were equal.

Another option is to use different hash functions.

It is quite obvious that the more hash functions or permutations we take the less would be the error of Jaccard similarity estimator. For any desired level of error the number of required repeats (permutations or hash functions) can be find out as follows [9]:

$$n=\frac{1}{\varepsilon^2},$$

where $n$ - is the number of required repeats,

$\varepsilon$ - desired level of error.

So, instead of dealing with huge matrices of user-user size we make $n$ computations with original user-item matrix and get sparse user-user similarity matrix.

**Conclusions and research prospects.** In the article we investigated typical collaborative filtering techniques, namely user-user CF and encountered that it is the subject of the following flaws:

1. In case of sparse data correlation doesn't work as it starts to account for majority of absent ranks rather than the present ones.
2. Switch to Cosine or Jaccard similarity measures solves the sparse problem but fits well only for binary ratings.
3. Extremely large number of users in the system makes it hard to handle user-user sized matrices even for modern distributed computing power.

To handle these flaws we proposed to substitute ranks with corresponding ranges of ranks that made Jaccard similarity measure more adequate and estimate Jaccard similarities using minhash trick that is supposed to be more efficient for extremely large datasets.

## References

1. *Harper, F. M. & Li, X. & Chen, Y. & Konstan, J. A.* (2005). An economic model of user rating in an online recommender system, in User Modeling 2005, vol. 3538 of Lecture Notes in Computer Science, Springer, August 2005, 307–316,

2. *Resnick, P. & Iacovou, N. & Suchak, M. Bergstrom, P. & Riedl, J.* (1994) GroupLens: an open architecture for collaborative filtering of netnews," in ACM CSCW '94, 175–186.

3. *Bennett J. & Lanning, S.* (2007). The netflix prize, in KDD Cup and Workshop'07.

4. *Herlocker, J. & Konstan, J. A. & Riedl, J.* (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," Information Retrieval, vol. 5, no. 4, 287–310.

5. *Sarwar, B. M. & Karypis, G. & Konstan, J. A. & Riedl, J. T.* (2000). Application

of dimensionality reduction in recommender system — a case study," in WebKDD 2000.

6. *Brand, M.* (2003). Fast online SVD revisions for lightweight recommender systems, in SIAM International Conference on Data Mining, 37–46.

7. *Broder, Andrei Z.* (1997). On the resemblance and containment of documents", Compression and Complexity of Sequences: Proceedings, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997 (PDF), IEEE, 21–29.

8. *Broder, Andrei Z. & Charikar, Moses & Frieze, Alan M. & Mitzenmacher, Michael* (1998), Min-wise independent permutations, Proc. 30th ACM Symposium on Theory of Computing (STOC '98), New York, NY, USA: Association for Computing Machinery, 327–336

9. *Leskovec, J. & Rajaraman, A. & Ullman, J.D.* (2014). Mining of massive datasets. Cambridge University PressReferences , 476.

**Руденський Р. А., Руденська В. В.**

**Ймовірнісні методи обробки великих даних в моделях рекомендаційних систем на основі колаборативної фільтрації**

У статті досліджуються стан та перспективи реалізації рекомендаційних систем в умовах розвитку методів обробки великих даних. Розкриваються основні проблеми, що виникають при впровадженні колаборативної фільтрації для подібних систем, та пропонується підхід для вирішення цих проблем. Пропонований підхід засновано на ідеї розширення матриці власних користувальницьких рейтингів і реалізації алгоритму MinHash, що дозволяє оцінити міру подібності Жаккара.

*Ключові слова:* рекомендаційні системи, великі дані, колаборативна фільтрація, подібність Жаккара, MinHash.

**Руденский Р. А., Руденская В. В.**

**Вероятностные методы обработки больших данных в моделях рекомендательных систем на основе коллаборативной фильтрации**

В статье исследуются состояние и перспективы реализации рекомендательных систем в условиях развития методов обработки больших данных. Раскрываются основные проблемы, возникающие при внедрении колаборативних фильтрации для подобных систем, и предлагается подход для решения этих проблем. Предлагаемый подход основан на идее расширения матрицы собственных пользовательских рейтингов и реализации алгоритма MinHash, что позволяет оценить меру подобия Жаккара.

*Ключевые слова:* рекомендательные системы, большие данные, коллаборативная фильтрация, подобие Жаккара, MinHash.

**Рецензент:** Метеленко Н. Г. – доктор економічних наук, професор, завідувач кафедри «Фінанси, банківська справа та страхування» Запорізької державної інженерної академії, м. Запоріжжя, Україна.

**Reviewer:** Metelenko N. – Professor, Ph.D. in Economics, Head of Finance, Banking and Insurance Department Zaporizhzhya State Ingeneering Academy Zaporizhzhya, Ukraine.

.

*e-mail:* Natalya381508@rambler.ru

*Стаття подана 27.02.2017 р.*