Time Series Forecasting by Recommendation: An Empirical Analysis on Amazon Marketplace

Álvaro Gómez-Losada(✉) and Néstor Duch-Brown

Joint Research Centre, European Commission, Seville, Spain
{alvaro.gomez-losada,nestor.duch-brown}@ec.europa.eu

Abstract. This study proposes a forecasting methodology for univariate time series (TS) using a Recommender System (RS). The RS is built from a given TS as only input data and following an item-based Collaborative Filtering approach. A set of top-N values is recommended for this TS which represent the forecasts. The idea is to emulate RS elements (the users, items and ratings triple) from the TS. Two TS obtained from Italy's Amazon webpage were used to evaluate this methodology and very promising performance results were obtained, even the difficult environment chosen to conduct forecasting (short length and unevenly spaced TS). This performance is dependent on the similarity measure used and suffers from the same problems that other RSs (e.g., cold-start). However, this approach does not require high computational power to perform and its intuitive conception allows for being deployed with any programming language.

Keywords: Collaborative Filtering · Time series · Forecasting · Data science

1 Introduction

Broadly speaking, autocorrelation is the comparison of a time series (TS) with itself at a different time. Autocorrelation measures the linear relationship between lagged values of a TS and is central to numerous forecasting models that incorporate autoregression. In this study, this idea is borrowed and incorporated to a recommender system (RS) with a forecasting purpose.

RSs apply knowledge discovery techniques to the problem of helping users to find interesting items. Among the wide taxonomy of recommendation methods, Collaborative Filtering (CF) [1] is the most popular approach for RSs designs [2]. CF is based on a intuitive paradigm by which items are recommended to an user

The original version of this chapter was revised: It has been changed to open access under a CC BY 4.0 license and the copyright holder is now "The Author(s)". The book has also been updated with these changes. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-20485-3_42

W. Abramowicz and R. Corchuelo (Eds.): BIS 2019, LNBIP 353, pp. 45–54, 2019.
https://doi.org/10.1007/978-3-030-20485-3_4

looking at the preferences of the people this user trusts. *User-based* or *item-based* [3] recommendations are two common approaches for performing CF. The first evaluates the interest of a user for an item using the ratings for this item by other users, called neighbours, that have similar rating patterns. On the other hand, item-based CF considers that two items are similar if several users of the system have rated these items in a similar fashion [4]. In both cases, the first task when building a CF process is to represent the user-items interactions in the form of a rating matrix. The idea is that given a rating data by many users for many items it can be predicted a user's rating for an item not known to the user, or identify a set of N items that user will like the most (top-N recommendation problem). The latter is the approach followed in this study.

The goal of this study is to introduce a point forecasting methodology for univariate TS using a item-based CF framework. In particular, to study the behaviour of this methodology in short lenght and unvenly spaced TS. On one side, the distinct values of a TS are considered the space of users in the RS. On the other, items are represented by the distinct values of a lagged version of this TS. Ratings are obtained by studying the frequencies of co-occurrences of values from both TS. Basically, the forecast is produced after averaging the top-N set of recommended items (distinct values of the shifted TS) to a particular user (a given value in the original TS).

2 Related Work

TS forecasting has been incorporated into recommendation processes in several works to improve the users' experience in e-commerce sites. However, to the best of the authors' knowledge, the use of a RS framework as a tool for producing forecasts in TS is new in literature.

2.1 Item-Based CF Approach

This section describes the basic and notation of an standard item-based CF RS, with a focus on the approach followed in this study. Mostly, CF techniques use a database as input data in the form of a user-item matrix $\mathbf{R}$ of ratings (preferences). In a typical item-based scenario, there is a set of m users $\mathcal{U} = \{u_1, u_2, ..., u_m\}$, a set of n items $\mathcal{I} = \{i_1, i_2, ..., i_n\}$, and a user-item matrix $\mathbf{R} = (r_{jk}) \in \mathbb{R}^{m \times n}$, with r_{jk} representing the rating of the user u_j $(1 \leq j \leq m)$ for the item i_k $(1 \leq k \leq n)$. In $\mathbf{R}$, each row represents an user u_j, and each column represents an item i_k. Some filtering criteria may be applied by removing from $\mathbf{R}$ those r_{jk} entries below a predefined $b \in \mathbb{N}^+$ threshold. One of the novelties of this study is the creation of an $\mathbf{R}$ matrix from a TS, which is explained in the next Sect. 3. Basically, $\mathbf{R}$ is created after using the TS under study and its lagged copy, and considering them as the space of users and items, respectively. The rating values (r_{jk}) are obtained by cross-tabulating both series. Instead of studying their relation with an autocorrelation approach, the frequency of co-occurrence of values is considered. In order to clarify how the transformation of

a TS forecasting problem is adapted in this study using a RS, Table 1 identifies some assumed equivalences.

Table 1. Some equivalences used in this study to build the forecasting recommender system (RS) from a given time series (TS).

Concept	Symbol	RS equivalence
Set of distinct values in TS	$\mathcal{U}$	Set of users with cardinality m
Set of distinct values in TS shifted	$\mathcal{I}$	Set of items with cardinality n
Two distinct values in the TS	u_j, u_l	A pair of users
Two distinct values of the shifted TS	i_i, i_j	A pair of items
Number of times a distinct value in the TS and its shifted version co-occurs	r_{jk}	Rating of user u_j on item i_k
TS value to which perform a forecasting	u_a	Active user to which recommend an item

The model-building step begins with determining a similarity between pair of items from $\mathbf{R}$. Similarities are stored in a new matrix $\mathbf{S} = (s_{ij}) \in \mathbb{R}^{n \times n}$, where s_{ij} represent a similarity between items i and j ($1 \leq i, j \leq n$). s_{ij} is obtained after computing a similarity measure on those users who have rated i and j items. Sometimes, to compute the similarity is set a minimum number of customers that have selected the (i, j) pair. This quantity will be referred as the $\mathsf{c} \in \mathbb{N}_{>2}$ threshold. Traditionally, among the most commonly similarity measures used are the Pearson correlation, cosine, constraint Pearson correlation and mean squared differences [5]. In this study it will be used the Cosine and Pearson correlation measures, but also, the Otsuka-Ochiai coefficient, which is borrowed from Geosciences [8] and used by leading online retailers like Amazon [9].

Some notation follows at this stage of the modelling. The vector of ratings provided for item i is denoted by $\mathbf{r}_i$ and $\bar{r}_i$ is the average value of these ratings. The set of users who has rated the item i is denoted by $\mathcal{U}_i$, the item j by $\mathcal{U}_j$, and the set of users who have rated both by $\mathcal{U}_{ij}$.

Forecasting. The aim of item-based algorithm is to create recommendations for a user, called the active user $u_a \in \mathcal{U}$, by looking into the set of items this user has rated, $I_{u_a} \in \mathcal{I}$. For each item $i \in I_{u_a}$, just the k items wich are more similar are retained in a set $\mathcal{S}(i)$. Then, considering the rating that u_a has made also on items in $\mathcal{S}(i)$, a weighted prediction measure can be applied. This approach returns a series of estimated rating for items different from those in I_{u_a} that can be scored. Just the top ranked items are included in the list of N items to be recommended to u_a (top-N recommended list). The k and N values has to be decided by the experimenter. In this study, each active user (u_a) was randomly selected from $\mathcal{U}$ following a cross-validation scheme, which is explained next. To every u_a, a set of recommendable items is presented (the forecast). The space

of items to recommend is represented by the distinct values of the shifted TS. Since the aim is to provide a point forecast, the numerical values included in the top-N recommended list are averaged.

Evaluation of the Recommendation. The basic structure for offline evaluation of RS is based on the train-test setup common in machine learning [5,6]. A usual approach is to split users in two groups, the training and test sets of users ($\mathcal{U}_{train} \cup \mathcal{U}_{test} = \mathcal{U}$). Each user in $\mathcal{U}_{test}$ is considered to be an active user u_a. Item ratings of users in the test set are split into two parts, the *query set* and the *target set*. Once the RS is built on $\mathcal{U}_{train}$, the RS is provided with the query set as user history and the recommendation produced is validated against the target set. It is assumed that if a RS performs well in predicting the withheld items (target set), it will also perform well in finding good recommendations for unknown items [7]. Typical metrics for evaluation accuracy in RS are root mean square error (RMSE), mean absolute error (MAE) or other indirect functions to estimate the novelty of the recommendation (e.g., serendipity). The MAE quality measure was used in this study.

3 Creation of the Rating Matrix

This section describes the steps to transform a given TS (**TS**) in a matrix of user-item ratings (**R**). This should be considered the main contribution of this study. The remaining steps do not differs greatly from a traditional item-based approach beyond the necessary adaptations to the TS forecasting context.

Since the aim it is to emulate **R**, the first step is to generate a space of users and items from **TS**. Thus, **TS** values are rounded to the nearest integer which will be called $\mathbf{TS_0}$. The distinct (unique) values of $\mathbf{TS_0}$ are assumed to be $\mathcal{U}$.

The second step is setting up $\mathcal{I}$. For that, $\mathbf{TS_0}$ is shifted forward in time $h \in \mathbb{N}_{>0}$ time stamps. The new TS is called $\mathbf{TS_1}$. The value of h depends on the forecasting horizon intended. Now, the distinct values of $\mathbf{TS_1}$ set $\mathcal{I}$. Once $\mathcal{U}$ and $\mathcal{I}$ have been set, they are studied as a bivariate distribution of discrete random variables. The last step is to compute the joint (absolute) frequency distribution of $\mathcal{U}$ and $\mathcal{I}$. Thus, it is assumed that $n_{jk} \equiv r_{jk}$, where n_{jk} representes the frequency (co-occurrence) of the u_j value of $\mathbf{TS_0}$ and i_k value of $\mathbf{TS_1}$. These steps are summarized below.

Algorithm. Rating matrix creation from a Time Series

Input:
Univariate time series **TS**
Parameters:
h : intended forecasting horizon
b : chosen threshold value for removing less representative frequencies

Output:
Rating matrix $\mathbf{R}$

Procedure:
Round to the nearest integer **TS** values $\rightarrow \mathbf{TS_0}$
Shift $\mathbf{TS_0}$ values h time stamps forward in time $\rightarrow \mathbf{TS}_1$
Remove first h values from $\mathbf{TS_0}$ and last h values from $\mathbf{TS_1}$
Obtain $\mathbf{TS_0}$ distinct values $\rightarrow \mathcal{U}$
Obtain $\mathbf{TS_1}$ distinct values $\rightarrow \mathcal{I}$
Cross-tabulate $\mathcal{U}$ and $\mathcal{I} \rightarrow \mathbf{R}$
Remove r_{jk} entries $\leq b$
Return $\mathbf{R}$

Some Considerations. The followed approach experiences the same processing problems that a conventional item-based approach, namely, sparsity in $\mathbf{R}$ and *cold start* situations (user and items with low or inexistent numbers of ratings). In large RS from e-commerce retailers, usually $n \ll m$. However, under this approach $n \simeq m$.

4 Experimental Evaluation

This section describes the sequential steps followed in this study for creating a item-based CF RS with the purpose of TS forecasting.

4.1 Data Sets

Two short length TS were used to evaluate this methodology. These TS were obtained after scraping the best-selling products from the Italy's Amazon webpage between 5th April, 2018 and 14th December, 2018. The scraping process setting was sequential. It consisted of obtaining the price from the first to the last item of each category, and from the first category to the latest. The first TS (TS-1) was created after averaging the evolution of best-selling products' prices included in the *Amazon devices* category. The second TS (TS-2) represents the price change of the most dynamic best-selling product from this marketplace site. Italy's Amazon webpage has experienced changes during the eight month period of the crawling process due to commercial reasons. The main change is related to the number of best-selling products being showed for each category (e.g., 20, 50 or 100). This causes the period of the scraping cycles is not similar and the TS derived from this data (TS-1 and TS-2) are not equally spaced at time intervals. In this study, the data obtained in the scraping process will be considered as a sequence of time events. Therefore, it is worth to note that the values of TS-1 were obtained after averaging a different number of products (20, 50, or 100), according to the number of products shown by Amazon at different times. Their main characteristics are shown in Table 2.

Table 2. TS characteristics used in the methodology testing (P: percentile; min: minimum value, max: maximum value; in €).

Abbreviation	Length	min	P50	max	P75-P25	Distinct values (m)
TS-1	2169	13	84	142	14	82
TS-2	1224	7	17	36	8	22

4.2 Creation of the Rating Matrices

A rating matrix **R** was created for TS-1 and TS-2 according to the algorithm described in Sect. 3. As mentioned before, the different duration of scraping cycles causes unevenly spaced observations in the TS. Also, it represents an additional difficulty when setting a constant forecasting horizon (h) as described in the algorithm. Therefore, in this study, it will be necessary to assume that the forecasting horizon coincides with the duration of the scraping cycle, independently of its length in time. In practice, this means that the TS representing the items ($\mathbf{TS_1}$) was obtained after lagging one position forward in time with respect the original TS representing the users ($\mathbf{TS_0}$). After empirical observation, h approximately takes values 1 h, 2 h or 4 h depending on Amazon shows the 20, 50 or 100 best-selling products for each category, respectively. The lack of proportionality between the duration of scraping cycles and the number of best-selling product shown is explained by technical reasons in the crawling process (structure of the Amazon's webpage for each product affecting the depth of the scraping process). Those ratings with a value $b \leq 3$ were removed from the corresponding **R**.

4.3 Similarity Matrix Computation

Three symmetric functions were used in this study to calculate different **S** for TS-1 and TS-2. The Pearson correlation (1) and cosine (2) similarity functions are standards in the RS field. The third one, the Otsuka-Ochiai coefficient, incorporates a geometric mean in the denominator:

$$s_{\mathbf{1}}(i,j) = \frac{\sum_{u \in \mathcal{U}_{i,j}} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{i,j}} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in \mathcal{U}_{i,j}} (r_{u,j} - \bar{r}_j)^2}} \tag{1}$$

$$s_{\mathbf{2}}(i,j) = \frac{\mathbf{r}_i \bullet \mathbf{r}_j}{\|\mathbf{r}_i\|_2 \, \|\mathbf{r}_j\|_2} = \frac{\sum_{u \in \mathcal{U}_{i,j}} r_{u,i} \bullet r_{u,j}}{\sqrt{\sum_{u \in \mathcal{U}_{i,j}} r_{u,i}^2} \sqrt{\sum_{u \in \mathcal{U}_{i,j}} r_{u,j}^2}} \tag{2}$$

$$s_{\mathbf{3}}(i,j) = \frac{|\, \mathcal{U}_{ij} \,|}{\sqrt{|\, \mathcal{U}_i \,| \, |\, \mathcal{U}_j \,|}} \tag{3}$$

where $\bullet$, $\| \cdot \|_2$ and $| \cdot |$ denote the dot-product, l_2 norm, and cardinality of the set, respectively. These similarity measures were calculated when the minimum number of user rating a given items was $c \geq 3$, and set the value of $k = 3$.

4.4 Generation of the Top *N*-Recommendation

This step begins by looking at the set of items the active user u_a has rated, I_{u_a}. In particular, the interest is to predict ratings for those items $j \notin I_{u_a}$ rated by user u_a. Then estimated rating ($\hat{r}_{u_a,j}$) for a given item $j \notin I_{u_a}$ was calculated according to (4), where $\mathcal{S}(j)$ denotes the items rated by the user u_a most similar to item j:

$$\hat{r}_{u_a,j} = \frac{1}{\sum_{i \in \mathcal{S}(j)} s(i,j)} \sum_{i \in \mathcal{S}(j)} s(i,j)\ r_{u_a,i} \tag{4}$$

The value of $\hat{r}_{(u,j)}$ can be considered a score that is calculated for each item not in I_{u_a}. Finally, the highest scored items are included in the top-N recommended list.

4.5 Evaluation

The set of users ($\mathcal{U}$) was splitted following a 80:20 proportion for obtaining the training and test sets of users ($\mathcal{U}_{train}$ and $\mathcal{U}_{test}$), respectively. Every user in $\mathcal{U}_{test}$ (20% of distinct values in $\mathbf{TS}_0$) was considered an active user (u_a) to whom recommend a set of items. From the history of each u_a, again the proportion 80:20 proportion was used to obtain the *query* and *target* sets, respectively. The produced recommendation was validated against the target set of each u_a.

Evaluation Metric. The MAE value was obtained according to (5):

$$MAE = \frac{\sum_{i=1}^{n} \mid e_i \mid}{n} \tag{5}$$

where n represent the number of active users to whom a recommendation has been suggested. The N value to generate a top-N recommendation was set dynamically according to the length of items in the *target* set for each u_a. Thus, the value of e is the difference between the average value of the top-N recommended items and the average value of the items in the target set.

4.6 Performance Results

The MAE results for assessing the quality of the forecasting proposal is shown in Table 3, for the two analysed TS and three similarity measures studied.

It can be seen that Otsuka-Ochiai similarity (s_3) yields a better performance on both TS studied. It is necessary to remind the characteristics of the TS used for testing this methodology (TS-1 and TS-2). They are characterized by unevenly spaced observations in the TS, but also, the variable forecasting horizon provided by the data acquisition environment. This represents a very complex environment for performing forecasting. In the case of forecasting the average

Table 3. MAE performance on both TS and the different similarity measures (s_1, s_2 and s_3: Pearson correlation, cosine and Otsuka-Ochiai similarities, respectively), in €.

	TS-1			TS-2		
	s_1	s_2	s_3	s_1	s_2	s_3
MAE	6.2	6.5	5.5	6.0	3.2	3.0

price for a given category, or the price for a given product, the forecasting results could be considered an estimation of the trend of such prices (uptrend or downtrend) more than aiming to forecast an exact value of such prices. Other experiences have been accomplished to test this methodology with ozone (O_3) TS (results now shown). These experiences with conventional TS (evenly spaced observations and constant forecasting horizon) has yield a very good results in term of forecasting. Therefore, TS-1 and TS-2 should be considered to represent an extreme environment in which perform forecasting. One of the advantage of this approach is that does not require high computational power to operate, due to the analysis of the distinct rounded values of a given TS, which is independent of its length. Besides, the proposed approach is very intuitive and allows for further developments to be included and being deployed using any programming language.

4.7 Computational Implementation

The computational implementation was accomplished using *ad hoc* designed Python functions except those specified in Table 4. The first two purposes correspond to the creation of the Rating matrix (**R**), and the remaining ones to the similarity matrix computation (**S**). In particular, the last two ones are used in the computation of the cosine similarity measure used in this work.

Table 4. Python functions for specific tasks.

Purpose	Function	Package
TS shifting	`shift`	pandas
Cross-tabulation	`crosstab`	pandas
To iterate over pairs of items	`itertools`	itertools
Dot product	`dot`	numpy
l_2 norm	`norm`	numpy.linalg

5 Conclusions

This study aims to produce a point forecast for a TS adopting a RS approach, in particular an item-based CF. To that end, basic elements in a RS (the

users, items and ratings triple) was emulated using a TS as only input data. An autocorrelation-based algorithm is introduced to create a recommendation matrix from a given TS. This methodology was tested using two TS obtained from Italy's Amazon webpage. Performance results are promising even the analyzed TS represent a very difficult setting in which to conduct forecasting. This is due to the TS are unevenly spaced and the forecasting horizon is not constant. Application of classical forecasting approaches (e.g., autoregression models) to this type of TS is not possible mainly due to the irregular time stamps in which observations are obtained. Thus, the introduced algorithm should be considered a contribution to the forecasting practice in both short length and unevenly spaced TS. Complementary, computational time required to obtain forecasting estimates is short due to such estimates are obtained considering distinct values of the TS are not all the values forming the TS. Further developments include to consider contextual information when building the RS and transforming the sequence of events to a TS with evenly spaced data.

Disclaimer. The views expressed are purely those of the authors and may not in any circumstances be regarded as stating an official position of the European Commission.

References

1. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM Spec. Issue Inf. Filter. **35**, 61–70 (1992). https://doi.org/10.1145/138859.138867
2. Sharma, R., Gopalani, D., Meena, Y.: Collaborative filtering-based recommender system: approaches and research challenges. In: 3rd International Conference on Computational Intelligence & Communication Technology, pp. 1–6. IEEE Press (2017). https://doi.org/10.1109/CIACT.2017.7977363
3. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, Hong Kong, 2001, pp. 285–295. ACM, New York (2001). https://doi.org/10.1145/371920.372071
4. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 107–144. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_4
5. Bobadilla, J., Ortega, F., Hernando, A., GutiéRrez, A.: Recommender Systems Survey. Knowl. Based Syst. **46**, 109–132 (2013). https://doi.org/10.1016/j.knosys.2013.03.012
6. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Found. Trends Hum. Comput. Interact. **4**(2), 81–173 (2010). https://doi.org/10.1561/1100000009
7. Haslher, M., Vereet, B.: recommenderlab: A Framework for Developing and Testing Recommendation Algorithms (2018). https://CRAN.R-project.org/package=recommenderlab
8. Ochiai, A.: Zoogeographical studies on the soleoid fishes found in Japan and its neighhouring regions-II. Bull. Japan. Soc. Sci. Fish **22**, 526–530 (1957). https://doi.org/10.2331/suisan.22.526

9. Jacobi, J.A., Benson, E.A., Linden, G.D.: Personalized recommendations of items represented within a database. US Patent US7113917B2 (to Amazon Technologies Inc.) (2006). https://patents.google.com/patent/US7113917B2/en
10. Breese, J.S, Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann Publishers (1998)