Exploiting the Evaluation Frequency of the Items to Enhance the Recommendation Accuracy

Roberto Saia, Ludovico Boratto, Salvatore Carta
Dipartimento di Matematica e Informatica
Università di Cagliari
Via Ospedale 72, 09124 - Cagliari, Italy
Email: roberto.saia@unica.it, ludovico.boratto@acm.org, salvatore@unica.it

***Abstract*—The main task of a recommender system is to suggest a list of items that users may be interested in. In this paper, we focus on the role that the *popularity* of the items plays in the recommendation process. If on the one hand, considering only the most popular items generates trivial recommendations, on the other hand, not taking in consideration the item popularity could lead to a non-optimal performance of a system, since it does not differentiate the items, giving them the same weight during the recommendation process. Therefore, we could risk to exclude from the recommendations some popular items that would have a high probability of being preferred by the users, suggesting instead others that, despite meeting the selection criteria, have less chance to be preferred. The proposed strategy aims to employ in the recommendation process new criteria based on the items' popularity, by introducing two novel metrics. Through the first metric we evaluate the semantic relevance of an item with respect to the user profile, while through the second metric, we measure how much it is preferred by users. Through a post-processing approach, we use these metrics in order to extend one of the most performing state-of-the-art recommendation techniques: SVD++. The effectiveness of this hybrid recommendation strategy has been verified through a series of experiments, which show strong improvements in terms of accuracy w.r.t. SVD++.**

Keywords*—*Semantic Analysis, Collaborative Filtering, Popularity, Metrics.

I. Introduction

In order to lead the potential buyers toward a number of well-targeted suggestions, related to the large amount of goods or services, a *Recommender System* (RS) plays a determinant role, since it is able to investigate on the user preferences, suggesting to users the items that could be interesting. In order to identify these items, a RS has to *predict* that an item is worth recommending [1]. Most of the strategies used to generate the recommendations are based on the so-called *Collaborative Filtering* (CF) approach [2], which is based on the assumption that users have similar preferences on a item if they have already rated other items in a similar way [3]. The rating prediction has been highlighted in the literature as the core recommendation task [1], and recent studies showed its effectiveness also in improving classification tasks [4], [5], [6]. In recent years, the *latent factor models* have been adopted in CF approaches with the aim to uncover latent characteristics that explain the observed ratings [7]. Some of the most common approaches of this type are those that exploit the *neural networks* [8], the *Latent Dirichlet Allocation* [9], but especially, those that exploit a model induced by the factorization of the user-item rating matrix [10] (i.e., the matrix that reports the ratings given to the items by the users). Among these last approaches, the state of the art is represented by SVD++ [11], the Koren's version of the *Singular Value Decomposition* (SVD) [12], which exploits the so-called *latent factor model* and presents good performance in terms of accuracy and scalability [13], [10]. Although SVD++ provides excellent performance, it does not take into account the factor of popularity of the items that are recommended, risking to penalize its performance under certain circumstances. This can happen when the same score is given to multiple items, since not being able to discriminate them on the basis of their popularity, there is the risk to recommend those unpopular, which are less likely to be preferred by the users.

The popularity of the items is an aspect that has been widely studied in the recommender systems literature. While their ability to identify items of potential interest to the users has been recognized, some limitations have been highlighted. The most important of these is that the recommendations made according to popularity criteria are trivial, and do not bring considerable benefits neither to users, nor to those that offer them goods or services. This happens when a system employs the so-called *non-personalized model* [14], a naive approach of recommendation that does not take into account the user preferences, because it always recommends a fixed list with the most popular items, regardless of the target user. On the other hand, however, recommending less popular items adds novelty (and also serendipity) [15] to the users, but usually it is a more difficult task to perform.

Another possible limitation that might occur when producing recommendations considering only the ratings is the fact that these approaches ignore the semantic relations between the words in the item descriptions. Therefore, thanks to the advent of the so-called Semantic Web [16], other strategies, based on semantic criteria [17], [18], have also spread. The main advantage is their capability to interpret the user preferences in a non-schematic mode, helping to understand the concepts that are connected with a text, which can be used to determine the similarity between items, instead of merely using the single terms in their textual description. In the literature it has been shown that some filtering methodologies can be used to pre-process the user profiles, in order to remove from them some useless items, increasing the accuracy of a recommender system [19].

This work aims at improving the recommendations produced by the SDV++ approach, by considering also the semantics behind the items and the items' popularity. This is

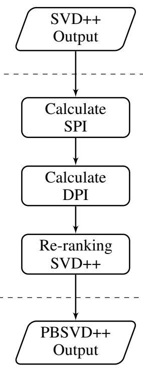


Fig. 1: Approach Architecture

done by employing two different strategies. The first strategy involves a balanced use of two indices of item popularity: one based on the positive feedbacks of the users, and one based on the conceptual similarity of the textual description of an item with the descriptions of the other ones positively evaluated in the past. The second strategy consists in the application of these two metrics within the boundaries of a recommendation list, generated through a state-of-the-art approach based on the *latent factor model* (the so-called SVD++ approach [11]), instead of using the entire dataset. This way of proceeding allows us to exploit the popularity metrics to perform a *fine-tuning* of the recommendations generated by a strategy at the state of the art, which does not take into account the items popularity, by improving the effectiveness of the generated recommendations. In conclusion, the proposed metrics enhance the performance of SVD++, since they are able to consider the popularity factor during its ranking process, giving priority to the items that have a high probability of being preferred by the users.

The contributions of our work are the following:

- definition of the Semantic Popularity Index (SPI), a metric able to evaluate the semantic popularity of an item, relatively to the items in a user profile;
- definition of the Domain Popularity Index (DPI), a metric able to evaluate the preferences of the users about an item;
- creation of the PBSVD++ algorithm, which extends the capabilities of SVD++, adding to it the capability to evaluate the item popularity.

The block diagram in Fig. 1 introduces the high level architecture of our approach. In the rest of this paper, we first introduce the literature related with the proposed strategy (Section II), continuing to define the adopted notation and the problem definition (Section III), the item popularity criteria (Section IV), and the implementation details of the PBSVD++ strategy (Section V). Finally, we complete the paper with the description of the performed experiments (Section VI), ending with some concluding remarks (Section VII).

II. Related Work

The recommender systems based on the so-called *non-personalized models* [14], propose to all users the same list of recommendations, without taking into account their preferences. This static approach is usually based on two algorithms, the first of them (TopPop), operates by suggesting the most rated items (i.e., those most popular), while the second (MovieAvg), works by suggesting the highest rated items (i.e., those most liked). The exclusive use of the non-personalized models, leads toward the absence of two important characteristic that a recommender system should have, i.e., novelty and serendipity [20]. Novelty occurs when a system is able to recommend unknown items that a user might have autonomously found, while serendipity happens when it helps the user to find a surprisingly interesting item that a user might not have otherwise found, or that it is very hard to find.

The type of data with which a recommender system operates is typically a sparse matrix where the rows represent the users, and the columns represent the items. The entries of this matrix are the interactions between users and items, in the form of ratings or purchases. The aim of a recommender system is to infer, for each user u, a ranked list of items, and in literature many of them are focused on the rating prediction problem. The most effective strategies in this field exploit the so-called *latent factor models*, but especially, the *matrix factorization* techniques [10]. Other CF ranking-oriented approaches that extend the matrix factorization techniques, have been recently proposed, and most of them use a ranking oriented objective function, in order to learn the latent factors of users and items [21]. SVD++ [11], the Koren's version of the *Singular Value Decomposition* (SVD) [12], is today considered one of the best strategies in terms of accuracy and scalability. In [22], [23], [24], the problem of modeling semantically correlated items was tackled, but the authors consider a temporal correlation and not the one between the items and a user profile.

III. Notation and Problem Definition

The mathematical notation used in this work, and the problem statement, are recalled in the following.

A. Notation

We consider a set of users $U = \{u_1, \ldots, u_N\}$, a set of items $I = \{i_1, \ldots, i_M\}$, and a set V of values used to express the user preferences (e.g., $V = [1, 5]$ or $V = \{like, dislike\}$). The set of preferences expressed by the users is a ternary relation $P \subseteq U \times I \times V$. We denote as $P_+ \subseteq P$ the subset of preferences with a positive value (i.e., $P_+ = \{(u, i, v) \in P | v \geq \overline{v} \vee v = like\}$), where $\overline{v}$ indicates the mean value (in the previous example, $\overline{v} = 3$). Moreover, we denote as $I_+ = \{i \in I | \exists (u, i, v) \in P_+\}$ the set of items for which there is a positive preference, and as $np_{i,U} = |(u, i, v) \in P_+|, i \in I, \forall u \in U$ the number of positive preferences of the users in U for an item i. We also denote as $I_u = \{i \in I | \exists (u, i, v) \in P \wedge u \in U\}$ the set of items in the profile of a user u, and as $R_u = \{u \in U \wedge R \subseteq I\}$, the set of items i recommended to a user u. The set of items I without the items already evaluated by the user u (i.e., those in I_u) is denoted as $\hat{I}_u \subseteq I$. Let $BoW = \{t_1, \ldots, t_W\}$ be the bag of words used to describe the items in I; we define as $S = \{s_1, \ldots, s_W\}$ the set of synsets associated to BoW (that is, for each term used to describe an item, we consider the associated synsets), as sd_i the semantic description of i, and as $sd_{I,u}$ the semantic description of all items i in the profile of the user u. The set of semantic descriptions is denoted as $D = \{sd_1, \ldots, sd_M\}$ (we have a semantic description for each item, so $|D| = |I|$). The approach used to extract sd_i and $sd_{I,u}$ from d_i is described in detail in Section V.

B. Problem definition

We consider the function $f : U \times I \rightarrow V$, adopted to predict the ratings for the not evaluated items with the SVD++ recommender system. Our aim is to define, for each item, a Semantic Popularity Index $SPI(i,u)$, able to evaluate the semantic relevance of each item $i \in \hat{I}_u$ with respect to the user profile I_u, and a Domain Popularity Index $DPI(i)$ that represents the popularity of the item with respect to the others in the dataset (in terms of positive evaluations given by the users to it). By defining a combined score α that involves both popularity indexes, our objective is to generate a list of recommended items i^* such that:

$$i^* = \underset{j \in \hat{I}_u}{\operatorname{argmax}} f(u,j) + \alpha \tag{1}$$

IV. Items Popularity

In the following, we introduce and formalize the two popularity indexes employed in our approach.

Semantic Popularity Index. The Semantic Popularity Index (SPI) for an item $i \in I$, with $SPI \in [0,1]$, is calculated as shown in Formula 2, where sd_i denotes the set of synsets extracted from the description of an item i to evaluate, and $sd_{I,u}$ the set of synsets extracted from the description of the items I in the profile of the target user u. It measures the conceptual similarity between these sets, and represents the *precision* (Section VI-D1), calculated for the item in the context of the user profile. SPI represents an important indicator, since through it we can estimate the level of (semantic) similarity of an item with the user tastes, represented in terms of items positively evaluated in the past.

$$SPI(i,u) = \frac{|sd_i \cap sd_{I,u}|}{|sd_i|} \tag{2}$$

Domain Popularity Index. The value of the Domain Popularity Index (DPI) for an item $i \in I$, with $DPI \in [0,1]$, represents the number $np_{i,U}$ of positive preferences expressed by all users U for the item i. It is calculated as shown in Formula 3. DPI is also an important indicator, because it extends the local information provided by SPI (related to the single users), providing a global measure of the preferences expressed for an item by all users.

$$DPI(i,U) = \frac{np_{i,U}}{\sum\limits_{\forall j \in I} np_{j,U}} \tag{3}$$

V. Implementation

In this section we present the steps made to generate the recommendations based on the proposed *Popularity-based* SVD++ (PBSVD++) strategy, starting from the extraction of the WordNet[1] synsets related to the textual description of the involved items, and ending with the implementation of the novel algorithm. These operations can be grouped into two steps: *Text Preprocessing* and *PBSVD++ Algorithm Definition*. In the first step, we process the textual description of the items in order to remove the useless elements, before the subsequent operation of synset retrieving. In the second step, we define the PBSVD++ algorithm, through which we can alter the original ranking of the SVD++ recommendations, by employing the SPI and DPI criteria.

A. Text Preprocessing

Motivated by the fact that exploiting a taxonomy for categorization and classification purposes is an approach recognized in the literature [25], [26], [27], in order to calculate the semantic correlation between the items we decided to exploit the functionalities offered by the WordNet environment. Before extracting the WordNet synsets from the text that describes each item, we need to follow several preprocessing steps. The first step is to detect the correct *Part-Of-Speech* (POS) for each word in the text. In order to perform this task, we have used the *Stanford Log-linear Part-Of-Speech Tagger* [28]. In the second step we remove punctuation marks and *stop-words*, which represent noise in the semantic analysis. In the third step, after we have determined the lemma of each word using the Java API implementation for WordNet Searching JAWS[2], we perform the so-called word sense disambiguation, a process where the correct sense of each word is determined, which permits us to individuate the appropriate synset in a precise way. The best sense of each word in a sentence was found using the Java implementation of the adapted Lesk algorithm provided by the *Denmark Technical University* similarity application [29]. All the collected synsets form the set $S = \{s_1, \dots, s_W\}$ defined in Section III. The output of this step is the semantic disambiguation of the textual description of each item $i \in I$, denoted as sd_i. For each user, we also extract an additional vector $sd_{I,u}$, which contains all the synsets that characterize the items she/he positively evaluated.

B. PBSVD++ Algorithm

We exploit the SPI and DPI metrics (explained in Section IV), in order to modify the result of the SVD++ approach, in accord with these two parameters. These two metrics are implemented in the Algorithm 1, where we merge them in a unique value α, given by their product. Given a set of recommendations R_u, addressed to a user $u \in U$, the final rating $\rho_{i,u}$ assigned to each item $i \in R_u$ by our algorithm, is composed by the $rating_{i,u}$ calculated through the SVD++ approach, normalized in a continuous range from 0 to 1, and denoted as $STD(i,u)$, added to the product of the two indices SPI and DPI (also normalized in a continuous range from 0 to 1), as shown in Formula 4. The final rating assigned to an item is then in the range from 0 to 2.

$$\begin{aligned} \rho_{i,u} &= STD(i,u) + \left(\frac{SPI(i,u)}{\sum\limits_{\forall j \in R_u} SPI(j,u)} \cdot \frac{DPI(i,U)}{\sum\limits_{\forall j \in R_u} DPI(j,U)} \right) \\ with\ STD(i,u) &= \frac{rating_{i,u}}{\sum\limits_{\forall j \in R_u} rating_{j,u}} \end{aligned} \tag{4}$$

The new rating $\rho_{i,u}$, assigned to an item i for a user u, takes into account, in a balanced way, both its semantic and domain popularities, and this produces a substantial change in the canonical SVD++ ranking during the recommendation process, changing the performance of the recommender system.

Algorithm 1 implements the operations described above. It takes as input the training set s (used by the SVD++ approach,

[1] A lexical database that groups words into set of synonyms called synsets

[2] http://lyle.smu.edu/ tspell/jaws/index.html

in step 3, to build the latent factor model), the user u to whom address the recommendations, and the number n of these. After the number x of potential items to recommend to the user u has been set (step 2), we calculate through the standard SVD++ approach, for the user u, a set I of x recommendations based on the training set s (step 3). In the steps from 5 to 11, we select from I only the elements i that are candidates for the recommendations based on the proposed approach. They are those items in which a modification of the score, by adding to the original rating of SVD++ the value of α (parameter calculated in the step 14, whose value is in the range from 0 to 1), could alter the rank proposed by SVD++. For this reason, the candidates are only the items to which, adding at most 1, we get a value higher than that of the item with the maximum SVD++ score (i.e., the first element i_0). We use this process also to calculate (in steps 8 and 9) the sum of the SPI and DPI weights, related to all the items $i \in I$. Starting with this set R of candidate items, in the steps from 12 to 18, we alter the SVD++ score of each item $i \in I$, following Formula 4, after which we return a list L of n recommendations, composed by the items with the higher score.

Algorithm 1 PBSVD++

```
Require: $s$=Training set, $u$=User, $n$=Recommendations
Ensure: $L$ = List of $n$ recommendations
 1: procedure GETPBSVDRECS($s$,$u$,$n$)
 2:     x=GetNumOfNotEvaluatedItems($u$)
 3:     I=GetSvdRecs($s$,$u$,$x$)
 4:     t1=0, t2=0
 5:     for each $i$ in $I$ do
 6:         if $(SvdRating(i) + 1) > SvdRating(i_0)$ then
 7:             $R \leftarrow i$
 8:             $t1 += GetSPI(i)$
 9:             $t2 += GetDPI(i)$
10:         end if
11:     end for
12:     for each $r$ in $R$ do
13:         rating=(SvdRating($r$)/SumAllSvdRatings($R$))
14:         $\alpha = (GetSPI(r)/t1) \cdot (GetDPI(r)/t2)$
15:         SetNewRating($r$,$rating$+$\alpha$)
16:     end for
17:     $L = GetRecsDescOrdered(R, n)$
18:     Return $L$
19: end procedure
```

VI. Experiments

In this section, after the definition of the experimental environment and of the adopted datasets' characteristics, we describe the strategy and metrics adopted, concluding with the presentation and discussion of the experimental results.

A. Experimental Setup

The environment for this work is based on the Java language, with the support of Java API implementation for WordNet Searching (JAWS) to perform the semantic analysis, and the support of Apache Mahout[3] Java framework to implement the state-of-the-art approach that we compare our approach with.

B. Datasets

In order to evaluate the proposed strategy, we perform a series of experiments on three different real-world datasets, extracted by two standard benchmarks for recommender systems: Yahoo! Webscope R4[4] and Movielens 10M[5].

Yahoo! Webscope (R4). This dataset contains a large amount of data related to users preferences expressed by the Yahoo! Movies community that are rated on the base of two different scales, from 1 to 13 and from 1 to 5 (we use the latter). The training data is composed by $7,642$ users ($|U|$), $11,915$ movies/items ($|I|$), and $211,231$ ratings ($|P|$). All the users in the training set have rated at least 10 items and all items are rated by at least one user. The test data is composed by $2,309$ users, $2,380$ items, and $10,136$ ratings. There are no test users/items that do not also appear in the training data. All the users in the test set have rated at least one item and all items have been rated by at least one user. The items are classified in 20 different classes (genres), and it should be noted that an item may be classified with multiple classes.

Movielens 10M. The second dataset used in this work is composed by $71,567$ users ($|U|$), $10,681$ movies/items ($|I|$), and $10,000,054$ ratings ($|P|$). It was extracted at random from MovieLens (a movie recommendation website). All the users in the dataset had rated at least 20 movies, and each user is represented by a unique ID. The ratings of the items are based on a *5-star* scale, with *half-star* increments. In this dataset the items are classified in 18 different classes (movie genres), and also in this case each item may be classified with multiple classes (genres). Since the Movielens 10M dataset does not contain any textual description of the items, to obtain this information we used a file provided by the Webscope (R4) dataset, which contains a mapping from the movie IDs used in the dataset to the corresponding movie IDs and titles used in the MovieLens dataset. Using the script provided with the Movielens 10M dataset, we split up the whole dataset in two different datasets with exactly 10 ratings per user in the test set. Both training sets are composed by $69,878$ users ($|U|$), and $9,301,274$ ratings ($|P|$), with $10,667$ movies/items ($|I|$) in the first one, and $10,676$ movies/items ($|I|$) in the second one. Each test dataset contains $69,878$ users ($|U|$), and $698,780$ ratings ($|P|$), with $3,326$ movies/items ($|I|$) in the first one, and $5,724$ movies/items ($|I|$) in the second one. From each of these datasets, we take in account a subset of $20,000$ users.

C. Strategy

We compare the proposed recommendation strategy with the state-of-the-art approach SVD++. The Mahout framework, used to implement it, in addition to the training set requires two parameters: the number of target features and the number of training steps to run. The first parameter would be equivalent to the number of involved genres, thus we have set this value to 20 for the Yahoo dataset, and to 18 for the Movielens datasets. Regarding the second parameter, we use the value 15, as indicated in the reference paper of the SVD++ algorithm [11].

D. Metrics

This section presents the metrics used in the experiments.

[3] https://mahout.apache.org

[4] http://webscope.sandbox.yahoo.com

[5] http://grouplens.org/datasets/movielens/

1) F1-Measure: The performance measure adopted to evaluate our approach, comparing the set of recommendations generated by our strategy and the set of those generated by the canonical approach of recommendation with the real user preferences stored in the test set, is the *F1-Measure* [30] metrics. Given two sets X_u and Z_u, where X_u denotes the set of recommendations performed for a user u, and Z_u the set of the real choices of the user u in the testset, this metric is defined as shown in Equation (5).

$$\begin{aligned}&F1\text{-}Measure(X_u, Z_u) = 2\frac{(precision(X_u, Z_u) \cdot recall(X_u, Z_u))}{(precision(X_u, Z_u) + recall(X_u, Z_u))}\\&\text{with}\\&precision(X_u, Z_u) = \frac{|Z_u \cap X_u|}{|X_u|},\ \ recall(X_u, Z_u) = \frac{|Z_u \cap X_u|}{|Z_u|}\end{aligned} \tag{5}$$

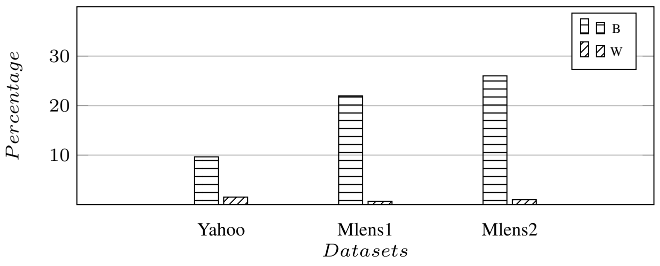
2) Metrics Evaluation: In order to compare the results of the two approaches of recommendation (i.e., our approach based on the PBSVD++ algorithm, and the canonical one, based on SVD++), we calculate the *F1-Measure* metric, presented in Equation (5), for each group of n performed recommendations (denoted as $@n$, with $n = \{2, 4, \ldots, 20\}$), subtracting from the values obtained by our approach those obtained by SVD++. In this way, a positive value denotes that our approach improves the standard one, while a negative value denotes that our approach worsens the standard one. A zero value means that the results are identical (i.e., proposed and standard approaches report the same performance). Denoting as X_n the set of n recommendations generated by our strategy, as Y_n the set of n recommendations generated by the canonical SVD++ strategy, and as Z_n the set of n real user preferences stored in the testset, we define the measure shown in Equation (6).

$$F1\text{-}variation@n = F1\text{-}Measure@n(X_n, Z_n) - F1\text{-}Measure@n(Y_n, Z_n) \tag{6}$$

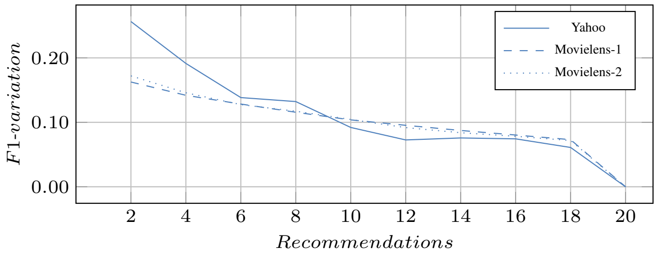
E. Experimental Results

Here, we report the results of the experiments.

1) Performance Overview and Details: The result presented in the part (a) of Figure 2 shows the general performance of the proposed strategy in the context of the three considered real-world datasets. It indicates the percentage of times in which we have done better, or have done worse than SVD++ (respectively, B and W). The overall results show the good performance of our approach with all three datasets. In the second set of experiments we compare the performance of a recommender system where we have implemented the PBSVD++ algorithm, with those of the canonic recommender system based on the SVD++ algorithm. We evaluate the results in terms of *F1-variation@n*, as described in Section VI-D. As we can observe in the part (b) of Figure 2, the results are quite similar for all three considered datasets. They show that our strategy outperforms the canonical one, except when we test the maximum number of recommendations (i.e., 20). This is an obvious aspect, since the algorithm PBSVD++ operates in the domain of the SVD++ recommendations, recalculating their ratings: therefore, when we consider the entire domain, the results of SVD++ and PBSVD++ are always identical.



(a) General performance



(b) F1-Measure@n

Fig. 2: Experiments Result

F. Discussion

The performed experiments, presented in Section VI-E1, prove that our strategy, based on the novel PBSVD++ algorithm, is able to improve the results of a canonical recommender system based on the SVD++ algorithm. As we can observe, this happens with any number of recommendations, except the case in which the maximum number of these is generated, for the obvious reason explained in the previous section. When evaluating these results, we can observe that the maximum value of positive variation for the metric is 1 (which represents a 100% improvement w.r.t. SVD++). Considering that we are confronted with a strategy of recommendation to the state of the art as SVD++, that offers a little margin of improvement, the results obtained can be considered highly satisfactory, also considering that we never did worse than SVD++. This proves that is possible to improve a state-of-the-art approach such as SVD++, by using its output as an input domain, in order to perform a *fine-tuning* based on the popularity of the involved items. Concluding, it should be noted that, although our approach outperforms SVD++ in the entire range of recommendations, it produces the best results with a few number of them. This represents an important aspect, considering the difficulty for a recommender system to make correct predictions, by generating few recommendations.

VII. Conclusions and Future Work

This paper proposed a new hybrid approach of recommendation, based on a novel algorithm called PBSVD++, which extends the state-of-the-art SVD++ strategy, adding to it the ability to evaluate two item popularity metrics. The performed experiments have shown both the validity of the adopted indexes, and their ability to improve the performance of the SVD++ approach. There is a possible application in a wide range of contexts, *in primis* those related to the recommender systems which operate in a commercial environment. In future work, we will extend our approach, by adding new metrics able to evaluate the item popularity, in the context of systems that operate within more than one domain of goods/services, trying to parametrize both the popularity aspect of each item, and their interconnections between different operative domains. We will also study the introduction of others metrics of popularity, e.g., based on the geographic or demographic information.

Acknowledgment

This work is partially funded by Regione Sardegna under project SocialGlue, through PIA - Pacchetti Integrati di Agevolazione “Industria Artigianato e Servizi” (annualità 2010), and by MIUR PRIN 2010-11 under project “Security Horizons”.

References

[1] F. Ricci, L. Rokach, and B. Shapira, “Introduction to recommender systems handbook,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2011, pp. 1–35.

[2] G. Karypis, “Evaluation of item-based top-n recommendation algorithms,” in *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*. ACM, 2001, pp. 247–254.

[3] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Adv. Artificial Intellegence*, vol. 2009, 2009.

[4] G. Armano and E. Vargiu, “A unifying view of contextual advertising and recommender systems,” in *KDIR 2010 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, A. L. N. Fred and J. Filipe, Eds. SciTePress, 2010, pp. 463–466.

[5] A. Addis, G. Armano, A. Giuliani, and E. Vargiu, “A recommender system based on a generic contextual advertising approach,” in *Proceedings of the 15th IEEE Symposium on Computers and Communications, ISCC 2010*. IEEE, 2010, pp. 859–861.

[6] E. Vargiu, A. Giuliani, and G. Armano, “Improving contextual advertising by adopting collaborative filtering,” *ACM Trans. Web*, vol. 7, no. 3, pp. 13:1–13:22, Sep. 2013.

[7] Y. Koren and R. M. Bell, “Advances in collaborative filtering,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2011, pp. 145–186.

[8] K. Georgiev and P. Nakov, “A non-iid framework for collaborative filtering with restricted boltzmann machines,” in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, ser. JMLR Proceedings, vol. 28. JMLR.org, 2013, pp. 1148–1156.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[10] Y. Koren, R. M. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[11] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Y. Li, B. Liu, and S. Sarawagi, Eds. ACM, 2008, pp. 426–434.

[12] M. J. P. Daniel Billsus, “Learning collaborative information filters,” in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, J. W. Shavlik, Ed. Morgan Kaufmann, 1998, pp. 46–54.

[13] J. Bennett, C. Elkan, B. Liu, P. Smyth, and D. Tikk, “Kdd cup and workshop 2007,” *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 51–52, Dec. 2007.

[14] P. Cremonesi, Y. Koren, and R. Turrin, “Performance of recommender algorithms on top-n recommendation tasks,” in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010*, X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, Eds. ACM, 2010, pp. 39–46.

[15] M. Ge, C. Delgado-Battenfeld, and D. Jannach, “Beyond accuracy: evaluating recommender systems by coverage and serendipity,” in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010*, X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, Eds. ACM, 2010, pp. 257–260.

[16] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, “The semantic web,” *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[17] T. Slimani, “Description and evaluation of semantic similarity measures approaches,” *CoRR*, vol. abs/1310.8059, 2013.

[18] R. Saia, L. Boratto, and S. Carta, “A latent semantic pattern recognition strategy for an untrivial targeted advertising,” in *Big Data (BigData Congress), 2015 IEEE International Congress on*. IEEE, 2015, pp. 491–498.

[19] R. Saia, L. Boratto, and S. Carta, “Semantic coherence-based user profile modeling in the recommender systems context,” in *Proceedings of the 6th International Conference on Knowledge Discovery and Information Retrieval, KDIR 2014*. SciTePress, 2014, pp. 154–161.

[20] L. Iaquinta, M. de Gemmis, P. Lops, G. Semeraro, M. Filannino, and P. Molino, “Introducing serendipity in a content-based recommender system,” in *8th International Conference on Hybrid Intelligent Systems (HIS 2008)*, F. Xhafa, F. Herrera, A. Abraham, M. Köppen, and J. M. Benítez, Eds. IEEE Computer Society, 2008, pp. 168–173.

[21] Y. Koren and J. Sill, “Ordrec: an ordinal model for predicting personalized item rating distributions,” in *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011*, B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, Eds. ACM, 2011, pp. 117–124.

[22] G. Stilo and P. Velardi, “Time makes sense: Event discovery in twitter using temporal similarity,” in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 02*, ser. WI-IAT ’14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 186–193.

[23] G. Stilo and P. Velardi, “Temporal semantics: Time-varying hashtag sense clustering,” in *Knowledge Engineering and Knowledge Management*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, vol. 8876, pp. 563–578.

[24] G. Stilo and P. Velardi, “Efficient temporal mining of micro-blog texts and its application to event discovery,” *Data Mining and Knowledge Discovery*, 2015.

[25] A. Addis, G. Armano, and E. Vargiu, “Assessing progressive filtering to perform hierarchical text categorization in presence of input imbalance,” in *KDIR 2010 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, A. L. N. Fred and J. Filipe, Eds. SciTePress, 2010, pp. 14–23.

[26] G. Armano, A. Giuliani, and E. Vargiu, “Semantic enrichment of contextual advertising by using concepts,” in *KDIR 2011 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, J. Filipe and A. L. N. Fred, Eds. SciTePress, 2011, pp. 232–237.

[27] G. Armano, A. Giuliani, and E. Vargiu, “Studying the impact of text summarization on contextual advertising,” in *2011 Database and Expert Systems Applications, DEXA, International Workshops*, F. Morvan, A. M. Tjoa, and R. Wagner, Eds. IEEE Computer Society, 2011, pp. 172–176.

[28] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, “Feature-rich part-of-speech tagging with a cyclic dependency network,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL ’03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 173–180.

[29] G. Salton, A. Wong, and C. S. Yang, “A vector space model for automatic indexing,” *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[30] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.