Factored Proximity Models for Top-N Recommendations

Athanasios N. Nikolakopoulos[1,3], Vassilis Kalantzis[2], Efstratios Gallopoulos[1] and John D. Garofalakis[1]
Department of Computer Engineering and Informatics[1], *University of Patras*
Department of Computer Science[2] *and Digital Technology Center*[3], *University of Minnesota*
{*anikolak,kalan019*}@*umn.edu*, {*stratis,garofala*}@*ceid.upatras.gr*

***Abstract*—In this paper, we propose EigenRec; a simple and versatile Latent Factor framework for Top-N Recommendations, which includes the well-known *PureSVD* algorithm as a special case. EigenRec builds a low dimensional model of an *inter-item proximity matrix* that combines a traditional similarity component, with a scaling operator, designed to regulate the effects of the prior item popularity on the final recommendation list. A comprehensive set of experiments on the *MovieLens* and the *Yahoo* datasets, based on widely applied performance metrics suggest that EigenRec outperforms several state-of-the-art algorithms, in terms of *Standard* and *Long-Tail* recommendation accuracy, while exhibiting low susceptibility to the problems caused by *Sparsity*, even its most extreme manifestations – the *Cold-start* problems.**

***Keywords*-Collaborative Filtering; Top-N Recommendation; Latent Factor Methods; PureSVD;**

I. Introduction

Collaborative Filtering (CF) is commonly regarded as one of the most effective approaches to building Recommender Systems (RS). Given a set of users, a set of items and – implicitly or explicitly – stated opinions about how much a user likes or dislikes the items he has already seen, CF techniques try to build "neighborhoods", based on the similarities between users (*user-oriented* CF) or items (*item-oriented* CF) as depicted from the data, in order to predict preference scores for the unknown user-item pairs, or provide a list of items that the user might find preferable.

Despite their success in real application settings, CF methods suffer several problems that remain to be resolved. One of the most significant such problems arises from the insufficiency of available data and is typically referred to as the *Sparsity* problem [1]. Sparsity is known to impose severe limitations to the quality of recommendations [2], and to decrease substantially the diversity and the effectiveness of CF methods – especially in recommending unpopular items (*Long-Tail* problem) [3]. Unfortunately, sparsity is an innate characteristic of recommender systems since in the majority of realistic applications, users interact with only a small percentage of the available items, with the problem being intensified even more, by the fact that newcomers with no ratings at all, are frequently added to the system (*Cold-Start* problem [2], [4]).

Although traditional CF techniques are very vulnerable to sparsity, *Graph-Based* methods manage to cope a lot better [1]. The fundamental characteristic that makes the methods of this family particularly suited for alleviating problems related to *limited coverage* and sparsity is that they allow elements of the dataset that are not directly connected to "influence" each other by propagating information along the edges of the graph [1]. Then, the transitive relations captured in this way, are used to recommend items either by estimating measures of proximity between the corresponding nodes [5] or by computing node similarity scores between them [6].

While promising in dealing with sparsity problems, the unprecedented growth of the number of users and listed items in modern e-commerce applications make many graph-based CF techniques suffer serious computational and scalability issues. *Latent Factor* methods, on the other hand, present a more viable alternative [1], [7]–[10]. The fundamental premise behind using latent factor models for building recommender systems is that user's preferences are influenced by a set of "hidden taste factors" that are usually very specific to the domain of recommendation [8]. These factors are generally not obvious and might not necessarily be intuitively understandable. Latent Factor algorithms, however, can infer those factors by the user's feedback as depicted in the rating data. Generally speaking, the methods in this family work by projecting the elements of the recommender database into a denser subspace that captures their most meaningful features, giving them the power to relate previously unrelated elements, and thus making them less susceptible to sparsity [1].

Summary of Contributions In this work we follow a latent factor-based approach and we propose a generic recommendation framework that combines computational efficiency with the necessary modeling freedom for achieving high-quality outcomes even in the presence of extreme sparsity. The method we propose – which we call EigenRec – works by building a low-dimensional subspace of a novel proximity matrix comprising *scaled inter-item similarity scores*. The pure similarity component can be defined utilizing any reasonable measure one sees fit for the recommendation problem under consideration; in this work, we make use of three simple similarity functions that were found to achieve very good performance in the movie and the song recommendation problems. The scaling component, on the other hand, allows fine-tuning the influence of the prior item popularity on the final proximity scores; a property that was found to enable our method to markedly improve the produced recommendation lists. To evaluate the performance of our method we conduct a comprehensive set of qualitative experiments on the `MovieLens` and `Yahoo` datasets and we show that EIGENREC produces recommendations that outperform several state-of-the-art methods in widely used metrics, achieving high-quality results even in the considerably harder task of recommending *Long-Tail* items. EIGENREC displays low sensitivity to the sparsity of the underlying space and shows promising potential in alleviating several related problems that occur commonly in recommender systems. This is true both in the very interesting case where sparsity is localized in a small part of the dataset – as in the *New Users* problem, and in the case where extreme levels of sparsity are found throughout the data – as in the *New Community* problem.

II. EIGENREC RECOMMENDATION FRAMEWORK

A. EigenRec Model Definitions

Let $\mathcal{U} = \{u_1, \dots, u_n\}$ be a set of *users* and $\mathcal{V} = \{v_1, \dots, v_m\}$ be a set of *items*. Let $\mathcal{R}$ be a set of tuples $t_{ij} = (u_i, v_j, r_{ij})$, where r_{ij} is a nonnegative number referred to as the *rating* given by user u_i to the item v_j, and let $\mathbf{R} \in \mathfrak{R}^{n\times m}$ be a matrix whose ij^{th} element contains the rating r_{ij} if the tuple t_{ij} belongs in $\mathcal{R}$, and zero otherwise. These ratings can either come from the explicit feedback of the user or inferred by the user's behavior and interaction with the system.

Inter-Item Proximity Matrix A. The Inter-Item Proximity matrix is designed to quantify the relations between the elements of the item space, as *properly scaled pure similarity scores*. Specifically, matrix $\mathbf{A} \in \mathfrak{R}^{m\times m}$ is a symmetric matrix, with its ij^{th} element given by:

$$A_{ij} \triangleq \xi(i,j)\cdot\kappa(i,j), \tag{1}$$

where $\xi(\cdot,\cdot) : \mathcal{V}\times\mathcal{V} \mapsto [0,\infty)$ is a symmetric *scaling function* and $\kappa(\cdot,\cdot) : \mathcal{V}\times\mathcal{V} \mapsto \mathfrak{R}$ is a symmetric *similarity function*.

Scaling Component. The definition of the scaling function can be done in many different ways, subject to various aspects of the recommendation problem at hand. In this work, we deploy this function as an easy way to regulate how much the inter-item proximity scores are affected by the prior popularity of the corresponding items. This was found to be very important for the overall recommendation quality as we will see in the experimental section of our paper. In particular, for the scaling function $\xi(\cdot,\cdot)$, we use the simple symmetric function

$$\xi(i,j) \triangleq (\|\mathbf{r_i}\|\|\mathbf{r_j}\|)^d. \tag{2}$$

where $\mathbf{r_i}$ denotes the i^{th} column of matrix $\mathbf{R}$. Notice that the definition of the scaling function allows the final inter-item proximity matrix to be written in factorial form:

$$\mathbf{A} = \mathbf{SKS} \tag{3}$$

where $\mathbf{S} \triangleq \operatorname{diag}\{\|\mathbf{r_1}\|, \|\mathbf{r_2}\|, \dots, \|\mathbf{r_m}\|\}^d$, and where matrix $\mathbf{K}$ (the ij^{th} element of which is defined to be $\kappa(i,j)$), denotes the pure similarity component.

Similarity Component. The definition of the similarity matrix $\mathbf{K}$ can be approached in several ways, depending on the nature of the recommendation task, the size of the itemset etc. Note that the final offline computational cost of the method may depend significantly on the choice of matrix $\mathbf{K}$ – especially when this matrix needs to be explicitly computed in advance or learned from the data. Having this in mind, in this work we propose using three widely used and simple similarity matrices that were found to be able to attain good results, while being easily manageable from a computational standpoint: (a) the *Cosine Similarity*, (c) the *Pearson-Correlation Similarity* and finally (c) the *Jaccard Similarity*.

Cosine Similarity $\mathbf{K_{COS}}$**.** The similarity function $\kappa(\cdot,\cdot)$ is defined to be the cosine of the angle between the vector representation of the items v_i, v_j, $K_{ij} \triangleq \cos(v_i, v_j)$.

Pearson Similarity $\mathbf{K_{PC}}$**.** The similarity score between two items v_i and v_j is defined as the ij^{th} element of matrix $\mathbf{K_{PC}}$ which is given by

$$K_{ij} \triangleq \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}}, \tag{4}$$

with C_{ij} denoting the covariance between the vector representation of the items v_i, v_j.

Jaccard Similarity $\mathbf{K_{JAC}}$**.** The Jaccard similarity between two items is defined as the ratio of the number of users that have rated both items to the number of users that have rated at least one of them. Specifically,

$$K_{ij} \triangleq \frac{|\mathcal{R}_i \cap \mathcal{R}_j|}{|\mathcal{R}_i \cup \mathcal{R}_j|}, \tag{5}$$

where $\mathcal{R}_i$ the set of users that have rated item i.

Recommendation Matrix $\mathbf{\Pi}$. The final recommendation matrix contains the recommendation vectors for each user in the system. In particular, for each user u_i the corresponding personalized recommendation vector is given by:

$$\boldsymbol{\pi}_\mathbf{i}^\intercal \triangleq \mathbf{r}_\mathbf{i}^\intercal\mathbf{V}\mathbf{V}^\intercal, \tag{6}$$

where $\mathbf{r}_\mathbf{i}^\intercal$ the ratings of user u_i and $\mathbf{V} \in \mathfrak{R}^{m\times f}$ is the matrix whose columns contain the f principal orthonormal eigenvectors of the inter-item proximity matrix $\mathbf{A}$. Observe that since $\mathbf{A}$ is real and symmetric, its eigenvectors are real and can be chosen to be orthogonal to each other and of unity norm.

B. Building the Latent Space

The specific properties of our model (symmetry and sparsity), allow us to use the symmetric *Lanczos algorithm* [11] – an iterative Krylov subspace method for solving symmetric eigenvalue problems – for building the latent space, $\mathbf{V}$, and producing the recommendation lists efficiently. The detailed computation of our recommendation matrix $\mathbf{\Pi}$ is given below.

Algorithm 1 EIGENREC

Input: Inter-Item proximity matrix $\mathbf{A} \in \mathfrak{R}^{m\times m}$. Rating Matrix $\mathbf{R} \in \mathfrak{R}^{n\times m}$. Latent Factors f.
Output: Matrix $\mathbf{\Pi} \in \mathfrak{R}^{n\times m}$ whose rows are the recommendation vectors for every user.

```
1: $\mathbf{q_j} = 0$, set $\mathbf{r} \leftarrow \mathbf{q}$ as a random vector
2: $\beta_0 \leftarrow \|\mathbf{r}\|_2$
3: for $j \leftarrow 1, 2, ...,$ do
4:     $\mathbf{q_j} \leftarrow \mathbf{r}/\beta_{j-1}$
5:     $\mathbf{r} \leftarrow \mathbf{A}\mathbf{q_j}$
6:     $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{q_{j-1}}\beta_{j-1}$
7:     $\alpha_j \leftarrow \mathbf{q_j^\intercal r}$
8:     $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{q_j}\alpha_j$
9:     $\mathbf{r} \leftarrow (\mathbf{I} - \mathbf{Q_j Q_j^\intercal})\mathbf{r}$,        ▷ where $\mathbf{Q_j} = [\mathbf{q_1}, \dots, \mathbf{q_j}]$
10:    $\beta_j \leftarrow \|\mathbf{r}\|_2$
11:    Solve the tridiagonal problem $(\mathbf{Q_j^\intercal A Q_j})\mathbf{\Xi_j} = \mathbf{\Theta_j \Xi_j}$
12:    Form the $j$ approximate eigenvectors $\mathbf{Q_j \Xi_j}$ of $\mathbf{A}$
13:    If the $f$ top eigenvectors have converged, stop.
14: end for
15: Compute latent factors $\mathbf{V} = \mathbf{Q_f \Xi}$
16: return $\mathbf{\Pi} \leftarrow \mathbf{RVV^\intercal}$
```

Computational Cost: The total cost introduced by the Matrix$\times$Vector (MV) products in j Lanczos steps amounts to $\mathcal{O}(j \cdot nnz)$, with nnz denoting the number of non-zero entries in matrix $\mathbf{A}$, whereas making the j^{th} Krylov vector orthogonal to the previous ones costs $\mathcal{O}(jm)$. If a large latent space must be built, the Lanczos procedure can be combined with polynomial filtering to accelerate the rate of convergence, e.g. see [12] for a related discussion and implementation.

C. PureSVD within EigenRec

A recent successful example of ranking-based latent factor recommendation algorithm is *PureSVD* [13]. This algorithm considers all missing values in the user-item rating matrix, $\mathbf{R}$, as zeros, and produces recommendations by estimating $\mathbf{R}$ by the factorization

$$\hat{\mathbf{R}} = \mathbf{U_f \Sigma_f Q_f^\intercal}, \tag{7}$$

where, $\mathbf{U_f}$ is an $n \times f$ orthonormal matrix, $\mathbf{Q_f}$ is an $m \times f$ orthonormal matrix, and $\mathbf{\Sigma_f}$ is an $f \times f$ diagonal matrix containing the f largest singular values. The rows of matrix $\hat{\mathbf{R}}$ contain the recommendation vectors for every user in the system. In what follows we will show that PureSVD is a simple member of the EigenRec family. In particular, let us consider the full singular value decomposition of $\mathbf{R}$:

$$\mathbf{R} = \mathbf{U\Sigma Q^\intercal}. \tag{8}$$

If we multiply equation (8) from the right with the orthonormal matrix $\mathbf{Q}$, we get $\mathbf{RQ} = \mathbf{U\Sigma}$. Now if we use $\mathbf{I_f}$ to denote the $f \times f$ identity matrix and we multiply again from the right with the $m \times m$ matrix $\left(\begin{smallmatrix}\mathbf{I_f} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right)$, we get

$$\mathbf{R}\left(\begin{smallmatrix}\mathbf{Q_f} & \mathbf{0}\end{smallmatrix}\right) = \mathbf{U}\left(\begin{smallmatrix}\mathbf{\Sigma_f} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{smallmatrix}\right) \quad \Rightarrow \quad \mathbf{RQ_f} = \mathbf{U_f\Sigma_f}. \tag{9}$$

Substituting equation (9) in (7) gives

$$\hat{\mathbf{R}} = \mathbf{RQ_f Q_f^\intercal}. \tag{10}$$

Relation (10) shows that the recommendation matrix of PureSVD can be expressed only in terms of the ratings matrix and matrix $\mathbf{Q_f}$ that contains the orthonormal set of eigenvectors that correspond to the f principal eigenvalues of the symmetric matrix

$$\mathbf{R^\intercal R} \equiv \begin{array}{cc} & \textit{users} \\ \textit{items} & \begin{bmatrix} — & \mathbf{r_i^\intercal} & — \end{bmatrix} \end{array} \times \begin{array}{cc} & \textit{items} \\ \textit{users} & \begin{bmatrix} | \\ \mathbf{r_j} \\ | \end{bmatrix} \end{array}$$

$$= \begin{array}{cc} & \textit{items} \\ \textit{items} & \begin{bmatrix} \boxed{\cdot} \end{bmatrix} \end{array} \quad \underbrace{\|\mathbf{r_i}\|\|\mathbf{r_j}\|}_{\textit{scaling}} \cdot \underbrace{\cos\theta_{ij}}_{\textit{similarity}}.$$

Therefore, the latent factor model of PureSVD is essentially built from the eigendecomposition of a scaled cosine-based inter-item similarity matrix; i.e. the final recommendation matrix of PureSVD coincides with that produced by our method, using the similarity matrix $\mathbf{K_{cos}}$ and the standard scaling matrix $\mathbf{S}$ with parameter $d = 1$. Seeing PureSVD within our framework hints that its default choice for the parameter d makes it overly sensitive to the prior popularity of the items and, as we will observe in our experimental section, it is exactly this suboptimal implicit choice of scaling that inevitably limits its quality.

Moreover, from a purely computational perspective, the above observation reduces the extraction of PureSVD's recommendation matrix to the calculation of the f principal eigenvectors of a symmetric matrix that can be expressed as a product of sparse factors; a fact that decreases significantly its overall computational and storage needs. To quantify the time difference that arises from the exploitation of EIGENREC's equivalent formulation for computing PureSVD, we run the two algorithms in MATLAB, on the `MovieLens10M` and `MovieLens20M` datasets [14], using in both cases MATLAB's native functions and the same convergence criterion. Figure 1 reports the results. As predicted, with our approach, PureSVD is computed significantly faster, with the speedup increasing with the dimension of the desired latent space ranging from 6 to 16 times faster computation for latent factors in the range $[50, \dots, 500]$.

III. EXPERIMENTAL EVALUATION

The recommendation quality of our method was tested using data originated from two recommendation domains, namely **Movie Recommendation** – where we exploit the

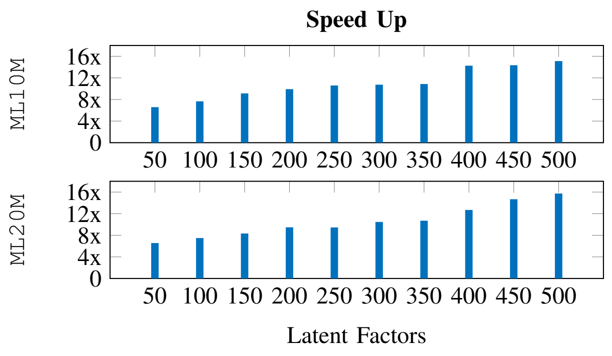

Figure 1. PureSVD's computation speed up using EIGENREC's formulation for latent factors in the range $[50, \ldots, 500]$.

standard MovieLens1M dataset [14] that has been used widely for the qualitative evaluation of recommender systems; and **Song Recommendation** – where we used the Yahoo!R2Music dataset [15] which represents a snapshot of the Yahoo!Music community's preferences for different songs. More details about the datasets can be found in [14], [15]. For our qualitative experiments, except for the standard **Recall** and **Precision** metrics we also use a number of well known *utility-based* measures, that have been proposed for the evaluation of the quality of top-N recommendation algorithms [16], namely the **Normalized Discounted Cumulative Gain (NDCG)** – which assumes that the utility of the recommendation is discounted *logarithmically fast* down the recommendation list; the **R-Score** – which assumes that the value of recommendations declines *exponentially fast*; and the **Mean Reciprocal Rank (MRR)** – which is the average of the reciprocal rank scores and assumes a slower decay than R-Score but faster than NDCG. For detailed definitions of the metrics due to space constrains we refer the reader to [16].

We compare EIGENREC[1] against a number of methods of the graph-based top-N recommendation family, that are considered to be highly promising in dealing with sparsity [1]. The five competing methods used in our experiments are: the *Pseudo-Inverse of the user-item graph Laplacian* (L†), the *Matrix Forest Algorithm* (MFA), the *Regularized Commute Time* (RCT), the *Markov Diffusion Kernel* (MD) and the *Relative Entropy Diffusion* (RED). For further details about the competing methods the reader should see [6] and the references therein.

A. Quality of Top-N Recommendations

For our recommendation quality comparison tests we used the complete MovieLens1M dataset (denoted ML1M) and – following the dataset preparation approach used by Karypis *et al.* in [18] – a randomly selected subset of the Yahoo! Research Alliance Webscope Dataset (denoted Yahoo) with 3312 items and 7307 users.

Except for the *Standard Recommendation*, we also test the performance of our method in dealing with two very challenging and realistic scenarios that are linked to the inherent sparsity of typical recommender systems datasets. Namely, the *Long-Tail Recommendation*, where we evaluate the ability of our method in making useful recommendations of unpopular items, and the *Cold-Start Recommendation*, where we evaluate how well it does in recommending items for *New Users* in an existing recommender system (localized sparsity) as well as making recommendations for a *New Community* of users in the starting stages of the system.

1) Standard Recommendations: To evaluate the quality of EIGENREC in suggesting top-N items, we have adopted the methodology proposed by Cremonesi *et al.* in [13]. In particular, we form a probe set $\mathcal{P}$ by randomly sampling 1.4% of the ratings of the dataset, and we use each item v_j, rated with 5-star by user u_i in $\mathcal{P}$ to create the test set $\mathcal{T}$. For each item in $\mathcal{T}$, we select randomly another 1000 unrated items of the same user, we rank the complete lists (containing 1001 items) using each of the competing methods, and we measure the respective recommendation quality.

First we test the recommendation performance of EIGENREC in the MRR metric for scaling parameters in the range $[-2, 2]$ using all three similarity matrices. We choose the MRR metric for this test simply because it can summarize the recommendation performance in a single number which allows direct comparisons between different similarity matrices as well as different scaling parameters for each given matrix. Figure 2 reports the MRR scores as a function of the parameter d for every case, using the number of latent factors that produces the best possible performance for each matrix.

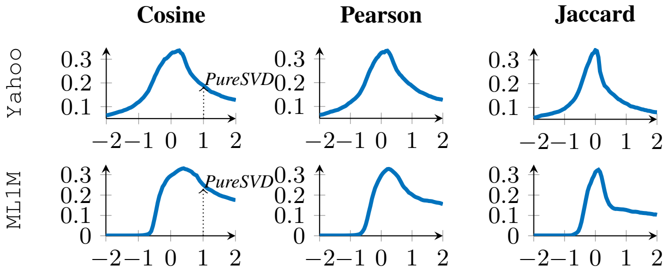

Figure 2. Recommendation performance of EIGENREC on the MRR metric for scaling factors in the range $[-2, 2]$ using all three similarity matrices.

We see that the best performance is achieved for small positive values of parameter d. This was true for every similarity matrix tested, and for both datasets. Notice that this parameter was included in our model as a means to control the sensitivity of the inter-item proximity scores to

[1]We make available a parallel implementation of our algorithm here: https://github.com/nikolakopoulos/EigenRec. For the implementation details as well as computational experiments see our extended technical report [17].

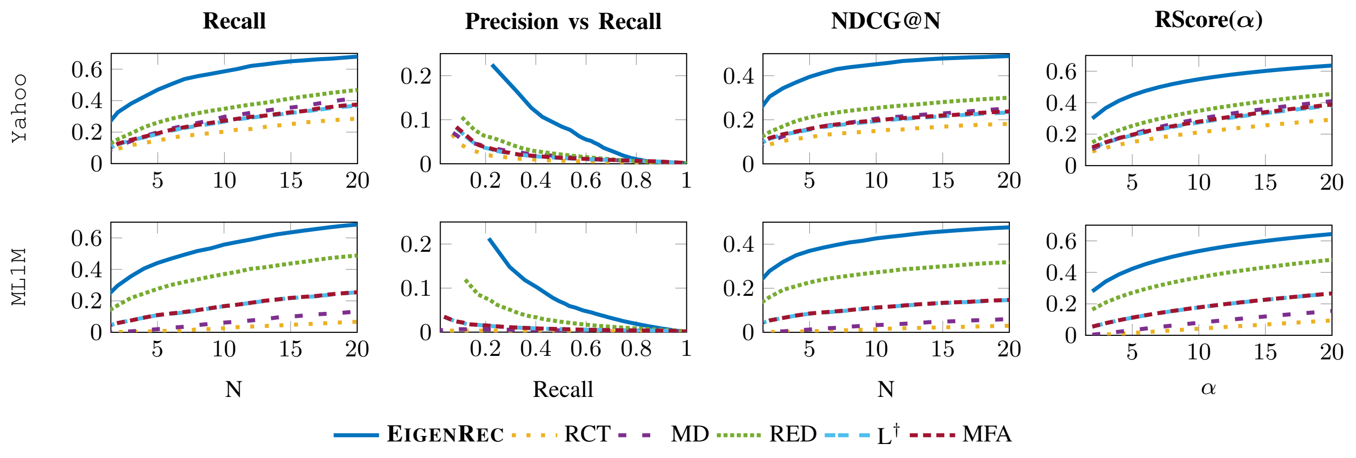

Figure 3. Evaluation of the recommendation quality using the Recall@N, Precision, NDCG@N and RScore metrics.

the prior popularity of the items under consideration. Our results suggest, that while this popularity is important (i.e. every time the best performing scaling factor was strictly positive), its contribution to the final matrix $\mathbf{A}$ should be weighted carefully so as not to overshadow the pure similarity component.

We see that all variations of our method outperform PureSVD every time, with the performance gap being significantly larger for the Yahoo dataset, which had a steeper performance decay as the scaling factors moved towards 1 (see Figure 2). Recall that the "black box" approach of the traditional PureSVD assumes cosine similarity (which is usually great) with scaling parameter d equal to 1 (which is usually not). As can be seen in Figure 2, simply controlling parameter d alone results to significant recommendation performance gains with respect to PureSVD. We find this particularly interesting, as it uncovers a fundamental limitation of the traditional PureSVD approach, that can be trivially alleviated with our approach.

We also compare EigenRec against the five graph-based methods mentioned in the beginning of this section. For these comparisons, we used the Jaccard similarity matrix. We tested each method for many different values of the parameters for every dataset and we report the best results achieved for each experiment. Figure 3 reports the Recall as a function of N (i.e. the number of items recommended) the Precision as a function of the Recall, the Normalized Discounted Cumulative Gain as a function of N and the RScore as a function of the halflife parameter α, for the `Yahoo` (first row) and the `MovieLens1M` (second row) datasets. As for Recall(N) and NDCG@N, we consider values of N in the range $[1, \dots, 20]$; larger values can be safely ignored for a typical top-N recommendation task [13]. As we can see, EIGENREC outperforms every other method considered, for all datasets and in all metrics, reaching for example, at $N = 10$ a recall around 60%. This means that 60% of the 5-starred items were presented in the top-10 out of the 1001 places in the recommendation lists of the respective users.

2) Long-Tail Recommendations: The distribution of rated items in recommender systems is long-tailed, i.e. most of the ratings are concentrated in a few very popular items, leaving the rest of the itemspace unevenly sparse. Of course, recommending popular items is an easy task, adding little utility in recommender systems; on the other hand, the task of recommending long-tail items adds *novelty* and *serendipity* to the users [13], and it is also known to increase substantially the profits of e-commerce companies [3], [19]. The innate sparsity of the problem however – which is aggravated even more for long-tail items – presents a major challenge for the majority of state-of-the-art collaborative filtering methods.

To evaluate EIGENREC in recommending long-tail items, we adopt the methodology described in [13]. In particular, we order the items according to their popularity which was measured in terms of number of ratings, and we partition the test set $\mathcal{T}$ into two subsets, $\mathcal{T}_{\text{tail}}$ and $\mathcal{T}_{\text{head}}$, that involve items originated from the long-tail, and the short-head of the distribution respectively. We discard the items in $\mathcal{T}_{\text{head}}$ and we evaluate EIGENREC and the other algorithms on the $\mathcal{T}_{\text{tail}}$ test set, using the procedure explained in Section III-A1.

Having evaluated the performance of EIGENREC in the MRR metric for all three similarity matrices, we obtained very good results for every case, with marginally better recommendation quality achieved for the Jaccard similarity component with 241 and 270 latent factors and scaling factor 0.2 and 0.4 for the `Yahoo` and the `MovieLens1M` datasets respectively. Proceeding with these parameter settings we run EIGENREC against the other graph-based algorithms and we report the results in Figure 4. It is interesting to notice that MFA and $L^\dagger$ do particularly well in the long-

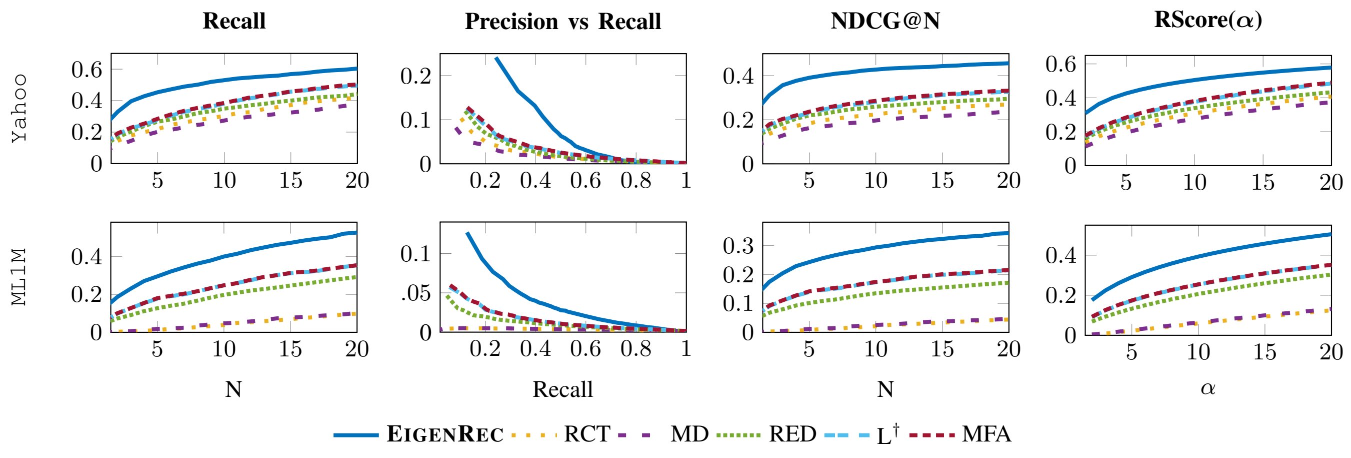

Figure 4. Evaluation of the Long-Tail recommendation quality using the Recall@N, Precision, NDCG@N and RScore metrics.

tail recommendation task, especially in the sparser Yahoo dataset. They even manage to surpass RED, which had reached the second place when the popular items were included (Figure 3). Once again, we see that EIGENREC achieves the best results, in all metrics and for both datasets.

We have seen that both in standard and long-tail recommendation scenarios, our approach gives very good results, consistently outperforming – besides PureSVD – a number of elaborate graph-based methods, known to work very well in uncovering nontrivial similarities through the exploitation of transitive relations that the graph representation of the data brings to light [1]. In our final set of experiments, presented next, we test the performance of EIGENREC in dealing with sparsity in its most extreme manifestations; the Cold-Start Problems.

3) Cold-Start Recommendations: The cold-start problem refers to the difficulty of making reliable recommendations due to an initial lack of ratings [2]. This is a very common problem faced by real recommender systems in their beginning stages, when the number of ratings for the collaborative filtering algorithms to uncover similarities between items or users are insufficient (*New Community Problem*). The problem can arise also when introducing new users to an existing system (*New Users Problem*); typically new users start with only a few ratings, making it difficult for the collaborative filtering algorithm to produce reliable personalized recommendations. This can be seen as a type of localized sparsity problem and it represents one of the ongoing challenges faced by recommender systems in operation.

New Community Problem: To test the performance of EIGENREC in dealing with the new community problem, we conduct the following experiment: We simulate the phenomenon by randomly selecting to include 33%, and 66% of the Yahoo dataset on two new artificially sparsified versions in such a way that the first dataset is a subset of the second. The idea is that these new datasets represent snapshots of the initial stages of the recommender system, when the community of users was new and the system was lacking ratings [20]. Then, we take the new community datasets and we create test sets following the methodology described in Section III-A1; we run all the algorithms and we evaluate their performance using the MRR, which makes it easier to compare the top-N quality for the different stages in the system's evolution. We test for both standard and long-tail recommendations and we report the results in Figure 5. We clearly see that EIGENREC outperforms every other algorithm, even in the extremely sparse initial stage where the system is lacking 2/3 of its ratings. In the figure, we report the qualitative results using the Cosine similarity this time, however, the performance of the three similarity components we propose was found to be equally good.

New Users Problem: In order to evaluate the performance of our algorithm in dealing with new users, we again use the Yahoo dataset and we run the following experiment. We randomly select 50 users having rated 100 items or more, and we randomly delete 95% of their ratings. The idea is that the modified data represent an "earlier version" of the dataset, when these users were new to the system, and as such, had fewer ratings. Then, we take the subset of the dataset corresponding to these new users and we create the test set as before, using 10% as a cut-off for the Probe Set this time, in order to have enough 5-rated movies in the Test Set to estimate reliably the performance quality. The results are presented in Figure 6. We see that EIGENREC manages to outperform all competing algorithms in all metrics as before.

4) Discussion: The qualitative results presented above indicate that our method is able to produce high quality recommendations, alleviating significant problems related to sparsity. Let us mention here that the competing algorithms

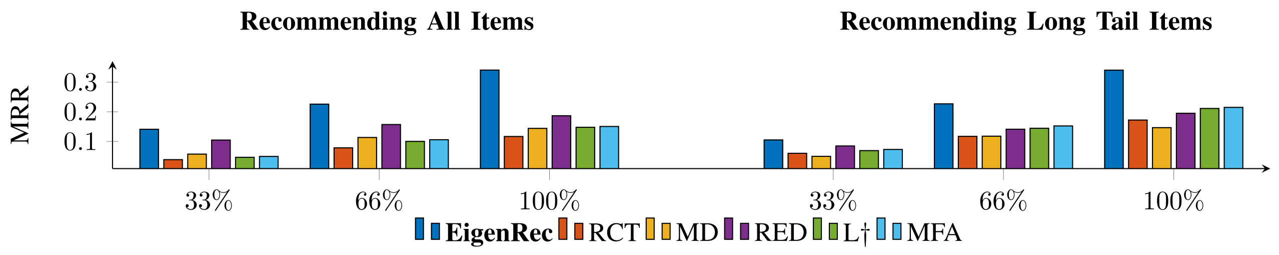

Figure 5. New-Community recommendation quality using the MRR metric.

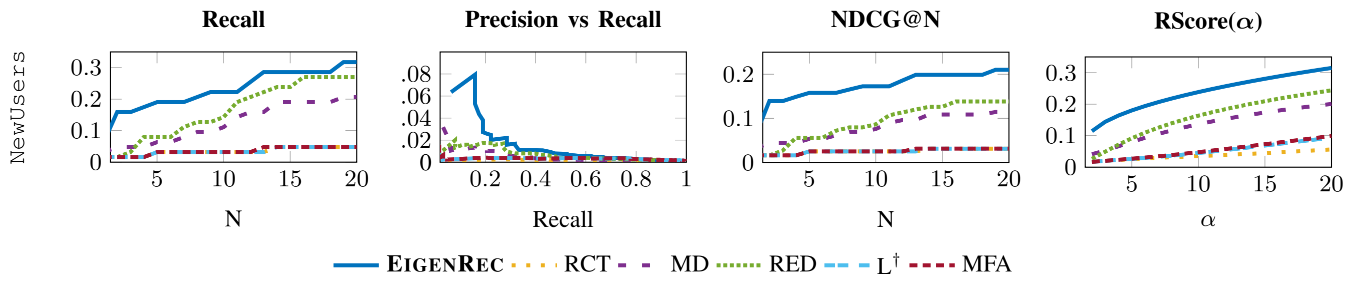

Figure 6. New-Users recommendation quality using the Recall@N, Precision, NDCG@N and RScore metrics.

are considered among the most promising methods in the literature to address sparsity problems [1]. This was verified in our experiments as well. Indeed, our results clearly show that the graph-based methods perform very well with their comparative performance increasing with the sparsity of the underlying dataset, and reaching its maximum in the cold-start scenarios. EigenRec nonetheless managed to perform even better, in every recommendation setting considered, being at the same time by far the most economical method from a computational point of view. Note here, all competing methods require handling a graph of $m + n$ nodes (where m the number of items and n the number of users), with the extraction of the recommendation scores many times involving inversions of $(m + n)$-dimensional square matrices etc. – problems that easily become intractable as the population of users in the system increases. EigenRec, on the contrary having a significantly friendlier computational profile, denotes a qualitative and feasible option for realistic top-N recommendation settings.

The choice of the scaling factor was found to be particularly significant for each and every pure similarity component. For the cosine similarity, in particular, we observed that the best results were always achieved for scaling parameters away from 1, making the traditional PureSVD algorithm, "qualitatively dominated" in every case considered. Regarding the best choice for the pure similarity component, the differences in recommendation quality observed in our experiments were relatively small. Therefore, our observations suggest that – at least for the recommendation scenarios considered in this work – all three simple inter-item proximity matrices present good candidates for high quality recommendations, with the $\mathbf{K}_{\text{cos}}$ being slightly more convenient to handle computationally.

IV. Remarks on Related Work

Factorization of a sparse similarity matrix was used to predict ratings of jokes in the EigenTaste system [9]. The authors first calculate the Pearson's correlation scores between the jokes and then form a denser latent space in which they cluster the users. The predicted rating of a user about a particular item is then calculated as the mean rating of this item, made by the rest of the users in the same cluster. The approach followed here differs significantly. The fact that we pursue ranking-based recommendations grants us the flexibility of not caring about the exact recommendation scores and allows us to introduce our novel proximity matrix, which except its pure similarity core also includes an important scaling component which was found to greatly influence the overall quality in every recommendation scenario.

The computational core of our method is the classic Lanczos algorithm that has been extensively used in the context of numerical linear algebra for the computation of the eigenvectors and/or singular triplets of large sparse matrices. From a qualitative perspective, Chen and Saad [21] have recently examined the use of Lanczos vectors in applications where the major task can be reduced to computing a matrix-vector product in the principal singular directions of the data matrix; they demonstrated the effectiveness of this approach on two different problems originated from information retrieval and face recognition. Also, in [22] the authors examine the use of Lanczos vectors for a very fast "crude" construction of a latent space that avoids overfitting extremely sparse datasets.

V. Conclusions

In this work, we proposed EigenRec; a versatile and computationally efficient latent factor framework for top-N recommendations; EigenRec works by building a low-dimensional subspace of a novel inter-item proximity matrix consisting of a similarity and a scaling component. We showed that the well-known PureSVD algorithm can be seen within our framework and we demonstrate experimentally that its implicit suboptimal treatment of the prior popularity of the items inevitably limits the quality of the recommendation lists it yields; a problem that can be painlessly alleviated through our approach.

An interesting direction that we are currently pursuing involves the definition of richer inter-item proximity matrices and the exploration of their effect in recommendation quality. In this paper, we restricted ourselves in using simple components that can be handled very efficiently from a computational point of view while being able to yield very good recommendations. We performed a comprehensive set of experiments on real datasets and we showed that EigenRec achieves very good results in widely used metrics against several state-of-the-art collaborative filtering techniques. Our method was also found to behave particularly well even when the sparsity of the dataset is severe – as in the New Community and the New Users versions of the Cold-Start problem – where it outperformed all other methods considered, including the very promising for their anti-sparsity properties graph-based techniques. In conclusion, our findings suggest that both the computational profile of EigenRec and its qualitative performance make it a promising candidate for the Standard, Long-Tail and Cold-Start recommendation tasks even in big data scenarios.

References

[1] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender Systems Handbook*. Springer US, 2011, pp. 107–144.

[2] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez, "Recommender systems survey," *Know.-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013.

[3] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, "Challenging the long tail recommendation," *Proceedings of the VLDB Endowment*, vol. 5, no. 9, pp. 896–907, 2012.

[4] A. N. Nikolakopoulos, M. A. Kouneli, and J. D. Garofalakis, "Hierarchical itemspace rank: Exploiting hierarchy to alleviate sparsity in ranking-based recommendation," *Neurocomputing*, vol. 163, pp. 126–136, 2015.

[5] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 116–142, Jan. 2004.

[6] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, and M. Saerens, "An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification," *Neural Networks*, vol. 31, pp. 53–72, 2012.

[7] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable collaborative filtering approaches for large recommender systems," *J. Mach. Learn. Res.*, vol. 10, pp. 623–656, Jun. 2009.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," DTIC Document, Tech. Rep., 2000.

[9] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[10] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 426–434.

[11] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office, 1950.

[12] J. L. Aurentz, V. Kalantzis, and Y. Saad, "cucheb: A gpu implementation of the filtered lanczos procedure," 2015.

[13] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proc. of the 4th ACM conf. on Recommender systems*, ser. RecSys '10, pp. 39–46.

[14] "GroupLens," http://grouplens.org/.

[15] "Yahoo Webscope," https://webscope.sandbox.yahoo.com/.

[16] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer US, 2011, pp. 257–297.

[17] A. N. Nikolakopoulos, V. Kalantzis, and J. D. Garofalakis, "Eigenrec: An efficient and scalable latent factor family for top-n recommendation," *arXiv preprint arXiv:1511.06033*, 2015.

[18] S. Kabbur, X. Ning, and G. Karypis, "Fism: Factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 659–667.

[19] C. Anderson, *The long tail: Why the future of business is selling less of more*. Hyperion, 2008.

[20] A. Nikolakopoulos and J. Garofalakis, "NCDREC: A decomposability inspired framework for top-n recommendation," in *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, vol. 1, Aug 2014, pp. 183–190.

[21] J. Chen and Y. Saad, "Lanczos vectors versus singular vectors for effective dimension reduction," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 8, pp. 1091–1103, 2009.

[22] A. N. Nikolakopoulos, M. Kalantzi, and J. D. Garofalakis, "On the use of lanczos vectors for efficient latent factor-based top-n recommendation," in *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, ser. WIMS '14. New York, NY, USA: ACM, 2014, pp. 28:1–28:6.