Overview

Tensor methods and recommender systems

Evgeny Frolov[1]* and Ivan Oseledets[1,2]

A substantial progress in development of new and efficient tensor factorization techniques has led to an extensive research of their applicability in recommender systems field. Tensor-based recommender models push the boundaries of traditional collaborative filtering techniques by taking into account a multifaceted nature of real environments, which allows to produce more accurate, situational (e.g., context-aware and criteria-driven) recommendations. Despite the promising results, tensor-based methods are poorly covered in existing recommender systems surveys. This survey aims to complement previous works and provide a comprehensive overview on the subject. To the best of our knowledge, this is the first attempt to consolidate studies from various application domains, which helps to get a notion of the current state of the field. We also provide a high level discussion of the future perspectives and directions for further improvement of tensor-based recommendation systems. © 2017 John Wiley & Sons, Ltd

How to cite this article:
WIREs Data Mining Knowl Discov 2017, 7:e1201. doi: 10.1002/widm.1201

INTRODUCTION

We live in the era of data explosion and information overload. Managing it would be impossible without the help of intelligent systems that can process and filter huge amounts of information much faster than humans. The need for such systems was already recognized in late 1970s in the Usenet, a distributed discussion platform, founded at Duke University. One of its goals was to help users to maintain numerous posts by grouping them into newsgroups. However, an active research on the topic of information filtering started in 1990s. The general term recommender systems (RS) was brought to the academia in the mid-90s with works of Resnick, Hill, Shardanand, and Maes[1] and was preceded by several famous projects: Tapestry, Lotus Notes, GroupLens.[2] A significant boost in RS research started after a famous Netflix prize competition with $1 million award for the winners, announced back in 2006. This has not only attracted a lot of attention from scientists and engineers, but also depicted the great interest from an industry.

Conventional RS deal with two major types of entities which are typically users (e.g., customers and consumers) and items (e.g., products and resources). Users interact with items by viewing or purchasing them, assigning ratings, leaving text reviews, placing likes or dislikes, and so on. These interactions, also called events or transactions, create an observation history, typically collected in a form of transaction/event log that reflects the relations between users and items. Recognizing and *learning* these relations in order to predict new possible interactions are the key goals of RS.

As we will see further, the definition of entities is not limited to users and items only. Entities can be practically of any type as long as predicting new interactions between them may bring a valuable knowledge and/or help to make better decisions. In some cases, entities can be even of the same type, like in the task of predicting new connections between people in a social network or recommending relevant paper citations for a scientific paper.

Modern recommender models may also have to deal with more than two types of entities within a

*Correspondence to: evgeny.frolov@skolkovotech.ru; ivan.oseledets@gmail.com

[1]Skolkovo Innovation Center, Skolkovo Institute of Science and Technology, Moscow, Russia

[2]Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow, Russia

Conflict of interest: The authors have declared no conflicts of interest for this article.

single system. For instance, users may want to assign tags (e.g., keywords) to the items they like. Tags become the third type of entity that relates to both users and items, as it represents the user motivation and clarifies items relevance (more on that in *Social Tagging* section). Time can be another example of an additional entity, as both user preferences and items relevance may depend on time (see *Temporal Models* section). Taking into account these multiple relations between several entities typically helps to provide more relevant, dynamic, and situational recommendations. It also increases complexity of RS models, which in turn brings new challenges and opens the door for new types of algorithms, such as tensor factorization (TF) methods.

The topic of building a production-ready RS is very broad and includes not only algorithms but also concerns a lot about business logic, dataflow design, integration with infrastructure, service delivery, and user experience. This also may require a specific domain knowledge and always needs a comprehensive evaluation. Speaking about the latter, the most appropriate way of assessing RS quality is an online A/B testing and massive user studies,[3–5] which are typically not available right at hand in academia. In this work, we will only touch mathematical and algorithmic aspects which will be accompanied with examples from various application domains.

The rest of the survey is divided into the following parts: *Recommender Systems at a Glance* and *Challenges for Recommender Systems* sections cover general concepts and major challenges in RS field; *Introduction to Tensors* section gives a brief introduction to tensor-related concepts, that are essential for understanding how tensors can be used in RS models; *Tensor-Based Models in Recommender Systems* section contains a comprehensive overview of various tensor-based techniques with examples from different domains; *Conclusion* section concludes the review and provides thoughts on possible future directions.

RS AT A GLANCE

Let us consider without loss of generality the task of product recommendations. The main goal of this task is, given some prior information about users and items (products), try to predict what particular items will be the most relevant to a selected user. The relevance is measured with some relevance score (or utility) function f_R that is estimated from the user feedbacks. More formally,

$$f_R : User \times Item \rightarrow Relevance\ \ Score, \tag{1}$$

where *User* is a domain of all users and *Item* is a domain of all items. The feedback can be either explicit or implicit, depending on whether it is directly provided by a user (e.g., ratings, likes/dislikes, etc.) or implicitly collected through an observation of his/her actions (e.g., page clicks, product purchases, etc.).

The type of prior information available in RS model defines what class of techniques will be used for building recommendations. If only an observation history of interactions can be accessed, then this is a task for collaborative filtering (CF) approach. If RS model uses intrinsic properties of items as well as profile attributes of users in order to find the best matching (*user* and *item*) pairs, then this is a content-based (CB) approach.

The complete overview of RS methods and challenges is out of scope of this survey and for a deeper introduction we refer the reader to.[3,6–8]

CB Filtering

As already mentioned, the general idea behind the CB approach is to use some prior knowledge about users preferences and items' properties in order to generate the most relevant recommendations. One of its main advantages is the ability to alleviate the cold start problem (see *Cold-Start* section) as long as all the needed content information is collected. Recommendations can be produced instantly even for those items that were never recommended to any user before.

This approach also has a number of issues, among which are the limited content analysis, overspecialization and high sensitivity to users input.[1,9] The key drawback from practical viewpoint is the difficulty of gathering descriptive and thorough properties of both items and users. This can be either manually performed by humans, i.e., with help of users and/or domain experts, or extracted automatically with data processing tools. The former method is usually very time consuming and requires considerable amount of work before RS can be built up. The latter method is highly dependent on information retrieval (IR) algorithms and is not always accurate or even possible.

Collaborative Filtering

In contrast to CB filtering, CF does not require any specific knowledge about users or items and only uses prior observations of users' collective behavior in order to build new recommendations. The class of CF techniques is generally divided into two categories: memory-based and model-based methods.[6,10]

Memory-Based CF

A widely used and very popular approach in this category is based on *k Nearest Neighbors* (kNN) algorithm.[11] It finds relevance scores for any (*user* and *item*) pair by calculating contributions from its neighbors. The neighborhood is typically determined by a similarity between either users (user-based approach) or items (item-based approach)[12] in terms of some similarity measure. This is also called a similarity-based approach. In its simplest implementation, the method requires to store in memory all prior information about user–item interactions in order to make predictions.

Performance of the similarity models may be greatly impacted by a selected measure of similarity (or a distance measure). Cosine similarity, Jaccard index, Pearson correlation, and Okapi BM25[13] are a few examples of possible choices. Even though the pure similarity-based models may give a good recommendations quality in some application domains, factorization models (see *Model-Based Collaborative Filtering* section) are better suited for large-scale problems often met in practice, delivering high-performance and high-quality recommendations.[6,14]

Model-Based CF

In the model-based approach, a predictive model is generated from a long enough history of observations and uses collective behavior of the crowd (a 'wisdom of crowds') in order to extract general behavioral patterns. One of the most successful model-based approaches is a matrix factorization (MF). The power of factorization models comes from the ability to embed users and items as vectors in a lower dimensional space of latent (or hidden) features (see *Dimensionality Reduction as a Learning Task* section). These models represent both users' preferences and corresponding items' features in a unified way so that the relevance score of the user–item interaction can be simply measured as an inner product of their vectors in the latent feature space.

As it follows from the description, both CF and CB tackle the problem of building relevant recommendations in very different ways and have their own sets of advantages and disadvantages. Many successful RS use *hybrid* approaches, that combine the advantages of both methods within a single model.[15,16]

CHALLENGES FOR RS

Building high-quality RS is a complex problem that involves not only a certain level of scientific knowledge but also greatly relies on an experience, passed from an industry and facing the real world implementations. This topic is also very broad and we will briefly discuss only the most common challenges that are closely related to an initial model design and its algorithmic implementations.

Cold-Start

Cold-start is the problem of handling new entities that concerns with both users and items.[3] When a new user is introduced to the system we usually know little or nothing about the user preferences and thus it makes it difficult or impossible to predict any interesting items for him or her. Similar problem arises when a new item appears in a product catalog. If an item has no content description or it was not rated by any user it will be impossible to build recommendations with this item.

Missing Values

Users typically engage with only a small subset of items and considerable amount of possible interactions stays unobserved. Excluding the trivial case of the lack of interest in specific items, there may be some other reasons for not interacting with them. For example, users may be simply unaware of existing alternatives for the items of their choice. Finding out those reasons helps to make better predictions and, of course, is a part of RS task. However, high level of uncertainty may bring an undesirable bias against unobserved data or even prevent RS models from learning representative patterns, resulting in low recommendations quality.

There are several commonly used techniques that help to alleviate these issues and improve RS quality. In MF case, simple regularization may prevent the undesired biases. Another effective technique is to assign some nonzero weights to the missing data, instead of completely ignoring it.[17] In hybrid models, a content information can be used in order to preprocess observations and assign nonzero relevance scores to some of the unobserved interactions (sometimes called as *sparsity smoothing*). These new data are then fed into standard CF procedure. Data clustering is another effective approach that is typically used to split the problem into a number of subproblems of smaller size with more connected information. Nevertheless, in case of a particular MF method, based on singular value decomposition

(SVD),[18] simply imputing zero relevance scores for an unobserved values may produce better results.[19,20] Additional smoothing can be achieved in that case with help of a *kernel trick*.[21] Other missing value imputation techniques based on various data averaging and normalization methods are also possible.[3] As we will see in *Tensor-Based Models in Recommender Systems* section, all of these techniques are valid in TF case as well.

Implicit Feedback

In many real systems users are not motivated or not technically equipped to provide any information about their actual experience after interacting with an item. Hence, user preferences can only be inferred from an implicit feedback, which may not necessarily reflect the actual user taste or even tell with guarantees whether the user likes an item or dislikes it.[17]

Model evaluation

Without a well-designed evaluation workflow and an adequate quality measures, it is impossible to build a reliable RS model that behaves equally well in both laboratory and production environments. Moreover, there are many aspects of a model assessment beyond recommendations accuracy that are related to both user experience and business goals. This can include metrics such as *coverage*, *diversity*, *novelty*, and *serendipity* (see Ref 22 for explanations) and indicators such as total generated revenue or average revenue per user session. This is still an open and ongoing research problem as it is not totally clear what are the most relevant and informative offline metrics and how to align them with the real online performance.

As mentioned in the beginning of *Challenges for Recommender Systems* section, the most reliable evaluation of RS performance is an online testing and user studies. Researchers typically do not have an access to a production systems so a number of offline metrics (mostly borrowed from IR field), became very popular. The most important among them are the relevance metrics: precision, recall, and F1-score; and the ranking metrics: normalized discounted cumulative gain (NDCG), mean average precision (MAP), mean reciprocal rank (MRR), and area under the ROC curve (AUC). These metrics may to some extent simulate a real environment, and in some cases have strong correlation with business metrics (e.g., recall and clickthrough rates (CTR)).[23]

It is also important to emphasize that while there are some real-world systems that target a direct prediction of a relevance score (e.g., rating), in most cases the main goal of RS is to build a good ranked list of items (top-n recommendations task). This imposes some constraints on the evaluation techniques and model construction. It might be tempting to use and optimize for error-based metrics like root-mean-squared error (RMSE) or mean absolute error (MAE) due to their simplicity. However, good performance in terms of RMSE does not guarantee a good performance on generating a ranked list of top-n recommendations.[3] In other words, the predicted relevance score may not align well with the perceived quality of recommendations.

Reproducible Results

The problem of reproducibility is closely related to recommendations quality evaluation. Careful design of evaluation procedures is critical for fair comparison of various methods. However, independent studies show that in controlled environments it is problematic to get consistent evaluation results even for the same algorithms on fixed datasets but within different platforms.[24]

Situation gets even worse, taking into account that many models, that tackle similar problems, use different datasets (sometimes not publicly available), different data preprocessing techniques,[25] or different evaluation metrics. In order to avoid unintended biases, we focus mostly on the description of the key features of existing methods rather than on a side-by-side comparison of quantitative results.

Real-Time Recommendations

A high-quality RS are expected not only to produce relevant recommendations but also respond instantly to the system updates, such as new (or unrecognized) users, new items, or new feedbacks.[14] Satisfying the latter requirement highly depends on the implementation: the predictive algorithms must have low computational complexity for producing new recommendations and take into account the dynamic nature of a real environments. Recomputation of the full RS model in order to include the new entities may take prohibitively long time and the user may never see a recommendation before he or she leaves. This means that RS application should be capable of making incremental updates and also be able to provide instant recommendations at a low computational cost outside of the full model recomputation loop. A number of techniques have been developed to fulfill these requirements for the MF case.[26–28] As shown in *Social Tagging* section, these ideas can be also applied in the TF case.

Incorporating Context Information

In the real world scenarios, interactions between users and items exhibit a multifaceted nature. User preferences are typically not fixed and may change with respect to a specific situation. For example, buyers may prefer different goods depending on the season of the year or time of the day. A user may prefer to watch different movies when alone or with a company of friends. We will informally call these situational aspects that shape user behavior, a contextual information or a context for short (see Figure 1). Other examples of context are location, day of week, mood, the type of a user's electronic device, and so on. Essentially, it can be almost anything.[29,30]

Context-aware RS (CARS) can be built with three distinct techniques:[31] contextual prefiltering, where a separate model is learned for every context type; contextual postfiltering, where adjustments are performed after a general context-unaware model was built; and contextual modeling, where context becomes an essential part of the training process. The first two techniques may lose information about the interrelations within a context itself. Contextual modeling, in turn, extends the dimensionality of the problem and promotes multirelational aspect into it. Therefore, it is likely to provide more accurate results.[32] Following Eq. (1), we can formalize it as follows:

$$f_R : User \times Item \times Context_1 \times \ldots \times Context_N \\ \rightarrow Relevance\ \ Score, \qquad (2)$$

where $Context_i$ denotes one of N contextual domains and the overall dimensionality of the model is $N + 2$.

As we will see further, TF models fit perfectly into the concept of CARS. With a very broad definition of context, tensor-based methods turn into a flexible tool, that allows to naturally model very interesting and nontrivial setups, where the concept of context goes beyond a typical connotation.

As a precaution, it should be noted that a nonspecificity of a context may lead to an interpretability problems. Using a general definition of a context, a content information such as user profile attributes (e.g., age and gender) or items properties (e.g., movie genre or product category) can also be regarded as some type of context (see, e.g., Ref 32 where age and gender are used to build new context dimensions). However, in practice, especially for TF models, this mixing is typically avoided.[33,34] One of the possible reasons is a deterministic nature of content information in contrast to what is usually denoted as a context. Similar to MF techniques, TF reveals new unseen associations (see *Dimensionality Reduction as a Learning Task* section) which in the case of deterministic attributes may be hard to interpret. It is easy to see in the following example.

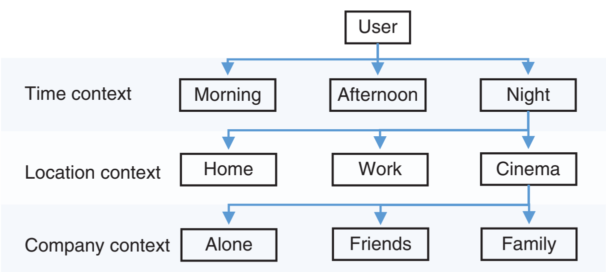

FIGURE 1 | Examples of contextual information.

For a triplet (*user*, *movie*, and *gender*), the movie rating may be associated with only one of two possible pairs of (*user* and *gender*), depending on the actual user's gender. However, once a reconstruction (with help of some TF technique) is made, a nonzero value of rating may now pop-up for both values of gender. The interpretation of such an association may become tricky and highly depends on initial problem formulation.

INTRODUCTION TO TENSORS

In this section, we only briefly introduce some general concepts needed for better understanding of further material. For a deeper introduction to the key mathematical aspects of multilinear algebra and TFs, we refer the reader to Refs 35–37. As in the case of MF in RS, TF produces a predictive model by revealing patterns from the data. The major advantage of a tensor-based approach is the ability to take into account a multifaceted nature of user–item interactions.

Definitions and Notations

We regard an array of numbers with more than two dimensions as a *tensor*. This is a natural extension of matrices to a higher order case. A tensor with m distinct dimensions or *modes* is called an m-way tensor or a tensor of order m.

Without loss of generality and for the sake of simplicity, we start our considerations with third-order tensors to illustrate some important concepts. We denote tensors with calligraphic capital letters, e.g., $\mathcal{T} \in \mathbb{R}^{M \times N \times K}$ stands for a third-order tensor of real numbers with dimensions of sizes M, N, K. We also use a compact form $\mathcal{T} = [t_{ijk}]_{i,j,k=1}^{M,N,K}$, where t_{ijk} is an element or entry at position (i, j, k), and assume everywhere in the text the values of the tensor to be real.

Tensor Fibers

A generalization of matrix rows and columns to a higher order case is called a *fiber*. Fiber represents a sequence of elements along a fixed mode when all but one indices are fixed. Thus, a mode-1 fiber of a tensor is equivalent to a matrix column, and a mode-2 fiber of a tensor corresponds to a matrix row. A mode-3 fiber in a tensor is also called a tube.

Tensor Slices

Another important concept is a tensor *slice*. Slices can be obtained by fixing all but two indices in a tensor, thus forming a two-dimensional array, i.e., matrix. In a third-order tensor, there could be three types of slices: horizontal, lateral, and frontal, which are denoted as $\mathcal{T}_{i::}, \mathcal{T}_{:j:}, \mathcal{T}_{::k}$, respectively.

Matricization

Matricization is a key term in TF techniques. This is a procedure of reshaping a tensor into a matrix. Sometimes it is also called unfolding or flattening. We follow the definition introduced in Ref 37. The n-mode matricization of a tensor $\mathcal{T} \in \mathbb{R}^{M \times N \times K}$ arranges the mode-n fibers to be the columns of the resulting matrix (see Figure 2). For the 1-mode matricization $T_{(1)}$ the resulting matrix size is $M \times (NK)$, for the 2-mode matricization $T_{(2)}$ the size is $N \times (MK)$, and the 3-mode matricization $T_{(3)}$ has the size $K \times (MN)$. In the general case of an mth-order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_m}$, the n-mode matricization $T_{(n)}$ will have the size $I_n \times (I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots I_m)$. For the corresponding index mapping rules, we refer the reader to Ref 37.

Diagonal Tensors

Another helpful concept is a diagonal tensor. A tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_m}$ is called diagonal if $t_{i_1 i_2 \ldots i_m} \neq 0$ only if $i_1 = i_2 = \ldots = i_m$. This concept helps to build a connection between different kinds of tensor decompositions.

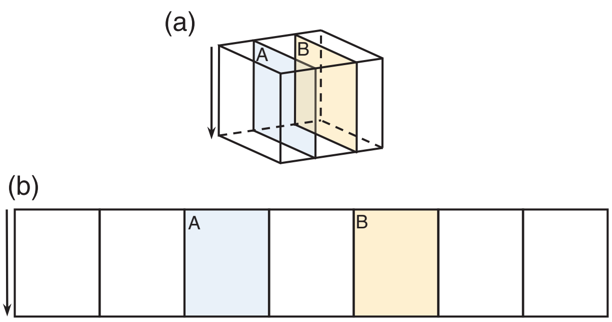

FIGURE 2 | Tensor of order 3 (a) and its unfolding (b). Arrow denotes the mode of matricization.

TF Techniques

The concept of TF can be better understood via an analogy with MF. For this reason, we first introduce a convenient notation and representation for the MF case and then generalize it to a higher order.

Dimensionality Reduction as a Learning Task

Let us first start with SVD, as it helps to illustrate some important concepts and also serves as a workhorse for certain TF techniques. Any matrix $A \in \mathbb{R}^{M \times N}$ can be represented in the form:

$$A = U \Sigma V^T, \tag{3}$$

where $U \in \mathbb{R}^{M \times K}$ and $V \in \mathbb{R}^{N \times K}$ are orthogonal matrices, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_K)$ is a diagonal matrix of nonnegative singular values $\sigma_1 \geq \ldots \geq \sigma_K$, and $K = \min(M, N)$ is a *rank* of SVD. According to the Eckart–Young theorem,[38] the truncated SVD of rank $r < K$ with $\sigma_{r+1}, \ldots, \sigma_K$ set to 0 gives the best rank-r (also called low-rank) approximation of matrix A. This has a number of important implications for RS models.

A typical user–item matrix in RS represents a snapshot of real 'noisy' data and it is practically never of low rank. However, the collective behavior has some common patterns which yield a low-rank structure and the real data can be modeled as:

$$R = A_r + E,$$

where E is a 'noise' and A_r is a rank-r approximation of data matrix R. The task of building a recommendation model translates into the task of recovering A_r (or equivalently, minimizing the noise E). For illustration purposes, here we assume that missing data in R is replaced with zeroes (other ways of dealing with missing values problem are briefly described in *Missing Values* section). Despite the simplicity of the assumption this approach is known to serve as a strong baseline.[19,20] The Eckart–Young theorem states, that an optimal solution to the resulting optimization task

$$\min_{A_r} \|R - A_r\|^2 \tag{4}$$

is given by the truncated SVD:

$$R \approx A_r = U_r \Sigma_r V_r^T. \quad (5)$$

Here and further in the text, we use $\|\cdot\|$ to denote Frobenius norm (both for matrix and tensor case), if not specified otherwise.

In terms of RS, factor matrices U_r and V_r, learned from observations, represent an embedding of users and items into the reduced latent space with r latent features. The dimensionality reduction produces a 'denoised picture' of data; it reveals a hidden (latent) structure that describes the relations between users and items. With this latent representation, some previously unobserved interactions can be uncovered and used to generate recommendations. This idea can be extended to a case of higher order relations between more than two entities and that is where the TF techniques come into play.

From now on, we omit the subscript r in the equations for both MF and TF, e.g., we denote a factor matrix U_r simply as U and likewise for other factor matrices. Let us also rewrite Eq. (3) in the form, that is useful for further generalization to higher order case:

$$A = \Sigma \times_1 U \times_2 V, \quad (6)$$

where $\times_n$ is an *n-mode product* which is typically defined for product between high order tensor and matrix. In matrix case, the n-mode product between two matrices A and B has the following form (assuming they are conformable):

$$(A \times_1 B)_{ij} = \sum_k a_{ki} b_{jk}, \quad (A \times_2 B)_{ij} = \sum_k a_{ik} b_{jk}.$$

In a more general case, each resulting element of an n-mode product between tensor $\mathcal{T}$ and matrix U is calculated as follows:[37]

$$(\mathcal{T} \times_n U)_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_m} = \sum_{i_n} t_{i_1 i_2 \ldots i_m} u_{j i_n}. \quad (7)$$

For the same purpose of further generalization, we rewrite Eq. (6) in two other forms, the index form:

$$a_{ij} = \sum_{\alpha=1}^{r} \sigma_\alpha u_{i\alpha} v_{j\alpha},$$

and a sum of rank-1 terms:

$$A = \sum_{\alpha=1}^{r} \sigma_\alpha \mathbf{u}_\alpha \otimes \mathbf{v}_\alpha, \quad (8)$$

where $\mathbf{u}_\alpha$, $\mathbf{v}_\alpha$ denote columns of the factor matrices, e.g., $U = [\mathbf{u}_1 \ldots \mathbf{u}_r]$, $V = [\mathbf{v}_1 \ldots \mathbf{v}_r]$ and $\otimes$ denotes the vector outer product (or dyadic product).

In the tensor case, we will also be interested in the task of learning a factor model from a real observations data $\mathcal{y}$. This turns into a dimensionality reduction problem that gives a suitable (not necessarily the best in terms of error-based metrics) approximation:

$$\mathcal{Y} \approx \mathcal{T},$$

where $\mathcal{T}$ is calculated with help of some of the tensor decomposition methods, described further. We will keep this notation throughout the text, e.g., $\mathcal{Y}$ will always be used to denote a real data and $\mathcal{T}$ will always be used to represent the reconstructed model, learned from $\mathcal{Y}$.

Candecomp/Parafac

The most straightforward way of extending SVD to higher orders is to add new factors in Eq. (8). In the third-order case, this has the following form:

$$\mathcal{T} = \sum_{\alpha=1}^{r} \lambda_\alpha \mathbf{u}_\alpha \otimes \mathbf{v}_\alpha \otimes \mathbf{w}_\alpha, \quad (9)$$

where each summation component $\mathbf{u}_\alpha \otimes \mathbf{v}_\alpha \otimes \mathbf{w}_\alpha$ is a *rank-1* tensor. We can also equivalently rewrite Eq. (9) in a more concise notation:

$$\mathcal{T} = [\![\lambda; U, V, W]\!], \quad (10)$$

where λ is a vector of length r with elements $\lambda_1 \geq \ldots \geq \lambda_r > 0$ and $U \in \mathbb{R}^{M \times r}$, $V \in \mathbb{R}^{N \times r}$, $W \in \mathbb{R}^{K \times r}$ defined similar to Eq. (8). The expression assumes that factors U, V, W are normalized. As we see further, in some cases values of λ can have a meaningful interpretation. However, in general, the assumption can be safely omitted, which yields:

$$\mathcal{T} = [\![U, V, W]\!] \equiv \sum_{\alpha=1}^{r} \mathbf{u}_\alpha \otimes \mathbf{v}_\alpha \otimes \mathbf{w}_\alpha, \quad (11)$$

or in the index from:

$$t_{ijk} = \sum_{\alpha=1}^{r} u_{i\alpha} v_{j\alpha} w_{k\alpha}. \quad (12)$$

The right-hand side of Eq. (11) gives an approximation of real observations data and is called Candecomp/Parafac (CP) decomposition of a tensor $\mathcal{Y}$.

Despite being similar to Eq. (8) formulation, there is a number of substantial differences in the concepts of tensor rank and low-rank approximation, thoroughly explained in Ref 37. Apart from technical considerations, an important conceptual difference is that there is no higher order extension of the Eckart–Young theorem (mentioned in the beginning of *Dimensionality Reduction as a Learning Task* section), i.e., if an exact low-rank decomposition of $\mathcal{Y}$ with rank r' is known, then its truncation to the first $r < r'$ terms may not give the best rank-r approximation. Moreover, the optimization task in terms of low-rank approximation is ill-posed,[39] which is likely to lead to numerical instabilities and issues with convergence, unless additional constraints on factor matrices (e.g., orthogonality, nonnegativity, etc.) are imposed.

Tucker Decomposition

A stable way of extending SVD to a higher order case is to transform the diagonal matrix Σ from Eq. (6) into a third-order tensor $\mathcal{G}$ and add an additional mode-3 tensor product (defined by Eq. (7)) with a new factor matrix W:

$$\mathcal{T} = [\![\mathcal{G}; U, V, W]\!] \equiv \mathcal{G} \times_1 U \times_2 V \times_3 W, \tag{13}$$

where $U \in \mathbb{R}^{M \times r_1}, V \in \mathbb{R}^{N \times r_2}, W \in \mathbb{R}^{K \times r_3}$ are orthogonal matrices, having similar meaning of the latent feature matrices as in the case of SVD. Tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is called a core tensor of the Tensor Decomposition (TD) and a tuple of numbers (r_1, r_2, r_3) is called a multilinear rank. The decomposition is not unique; however, the optimization problem with respect to multilinear rank is well-posed. Note, that if tensor $\mathcal{G}$ is diagonal with all ones on its diagonal, than the decomposition turns into CP. In the index notation, TD takes the following form:

$$t_{ijk} = \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} g_{\alpha\beta\gamma} u_{i\alpha} v_{j\beta} w_{k\gamma}. \tag{14}$$

The definition of TD is not restricted to have three modes only. Generally, the number of modes is not limited; however, storage requirements depend exponentially on the number of dimensions (see Table 1), which is often referred as a *curse of dimensionality*. This imposes strict limitations on the number of modes for many practical cases, whenever more than four entities are modeled in a multilinear way (e.g., *user*, *item*, *time*, *location*, *company*, or any other context variables; see Figure 1). In order to break the curse of dimensionality, a number of efficient methods has been developed recently, namely Tensor Train (TT)[40] and Hierarchical Tucker (HT).[41] However, we are not aware of any published results related to TT- or HT-based implementations in RS.

TABLE 1 | Storage Requirements for Different Tensor Factorization Methods

	CP	TD	TT	HT
Storage	dnr	$dnr + r^d$	dnr^2	$dnr + dr^3$

CP, Candecomp/Parafac; HT, Hierarchical Tucker; TD, Tensor Decomposition; TT, Tensor Train.
For the sake of simplicity, this assumes a tensor with d dimensions of equal size n and all ranks (or rank in case of CP) of a tensor decomposition set to r.

Optimization Algorithms

Let us start from the simplest form of an optimization task where the objective J is defined by a loss function L as follows:

$$J(\theta) = L(\mathcal{Y}, \mathcal{T}(\theta)), \tag{15}$$

where θ denotes model parameters, i.e., $\theta := (U, V, W)$ for CP-based models and $\theta := (\mathcal{G}, U, V, W)$ in case of TD. The optimization criteria takes the following form:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, J(\theta), \tag{16}$$

where θ^* defines an optimal set of model parameters which are to be used to generate recommendations.

It is convenient to distinguish the three general categories of optimization objectives that lead to different ranking mechanisms: *pointwise*, *pairwise*, and *listwise*.[42] Pointwise objective depends on a pointwise loss function between the observations y_{ijk} and the predicted values t_{ijk}. In case of a square loss, the total loss function has a similar to Eq. (4) form:

$$L(\mathcal{Y}, \mathcal{T}) = ||\mathcal{Y} - \mathcal{T}||^2. \tag{17}$$

Pairwise objective depends on a pairwise comparison of the predicted values and penalizes those cases when their ordering does not correspond to the ordering of observations. The total loss can take the following form in that case:

$$L(\mathcal{Y}, \mathcal{T}) = \sum_{i,j} \sum_{k,k' : y_{ijk} > y_{ijk'}} l\left(t_{ijk} - t_{ijk'}\right),$$

where $l(x)$ is a pairwise loss function that decreases with the increase of x.

Listwise objective operates over whole sets (or lists) of predictions and observations. The listwise loss function, that can be schematically expressed as $l(\{t_{ijk}\}, \{y_{ijk}\})$, penalizes the difference between the predicted ranking of a given list of items and the ground truth ranking, known from observations.

Pointwise Algorithms for TD

In case of TD-based model, the solution to Eq. (16) given Eq. (17) can be found with help of two well-known methods proposed in Ref 43: Higher-Order SVD (HOSVD)[44–46] or Higher-Order Orthogonal Iteration (HOOI).[47,48]

The HOSVD method can be described as a consecutive application of SVD to all three matricizations of $\mathcal{Y}$, i.e., $Y_{(1)}$, $Y_{(2)}$, $Y_{(3)}$ (assuming that missing data are imputed with zeros). Generally it produces a suboptimal solution to an optimization problem induced by Eq. (15); however, it is worse than the best possible solution only by a factor of $\sqrt{d}$, where d is the number of dimensions.[49] Due to its simplicity this method is often used in RS literature.

The HOOI method uses an iterative procedure based on an alternating least squares (ALS) technique, which successively optimizes Eq. (15). In practice, it may require a small amount of iterations to converge to an optimal solution, but in general it is not guaranteed to find a global optimum.[37] The choice of any of these two methods for particular problem may require additional investigation in terms of both computational efficiency and recommendations quality before the final decision is made.

The orthogonality constraints imposed by TD may in some cases have no specific interpretation. Relaxing these constraints leads to a different optimization scheme, typically based on gradient methods, such as stochastic gradient descent (SGD).[32] The objective in that case is expanded with a regularization term $\Omega(\theta)$:

$$J(\theta) = L(\mathcal{Y}, \mathcal{T}(\theta)) + \Omega(\theta), \tag{18}$$

which is commonly expressed as follows:

$$\Omega(\theta) = \lambda_G||\mathcal{G}||^2 + \lambda_U||U||^2 + \lambda_V||V||^2 + \lambda_W||W||^2, \tag{19}$$

where λ_G, λ_U, λ_V, λ_W are regularization parameters and usually $\lambda_U = \lambda_V = \lambda_W$.

Pointwise Algorithms for CP

As has been noted in *Candecomp/Parafac* section, CP is generally ill-posed and if no specific domain knowledge could be employed to impose additional constraints, a common approach to alleviate the problem is to introduce regularization similar to Eq. (19):

$$\Omega(\theta) = \lambda_U||U||^2 + \lambda_V||V||^2 + \lambda_W||W||^2, \tag{20}$$

Indeed, depending on the problem formulation it may also have more complex form both for CP (e.g., as in *BPTF* section) and TD models. In general, regularization allows to ensure convergence and avoid degeneracy (e.g., when rank-1 terms become close to each other by absolute value but their magnitudes go to infinity and have opposite signs);[37] however, it may lead to a sluggish rate of convergence.[50] In practice, however, many problems can still be solved with CP using variations of both ALS[23,51] and gradient-based methods.

Pairwise and Listwise Algorithms

Pairwise and listwise methods are considered to be more advanced and accurate as they are specifically designed to solve ranking problems. The objective function is often derived directly from a definition of some ranking measure, e.g., pairwise AUC or listwise MAP (see Refs 52 and 53 for CP-based and TD-based implementations respectively), or constructed in a way that is closely related to those measures.[33,54]

These methods typically have a nontrivial loss function with complex data interconections within it which makes it hard to optimize and tune. In practice, the complexity problem is often resolved with help of handcrafted heuristics and problem-specific constraints (see *RTF and PITF* and *TFMAP* sections), which simplify the model and improve computational performance.

TENSOR-BASED MODELS IN RS

Treating data as tensor may bring new levels of flexibility and/or quality into RS models; however, there are nuances that should be taken into account and treated properly. This section covers different tensorization techniques used to build advanced RS in various application domains. For all the examples, we use a unified notation (where it is possible) introduced in *Introduction to Tensors* section, hence it might look different from the notation used in the original papers. This helps to reuse some concepts within different models and build a consistent narrative throughout the text.

Personalized Search and Resource Recommendations

There is a very tight connection between personalized search and RS. Essentially, recommendations can be considered as a *zero query search*[55] and, in turn, personalized search engine can be regarded as a query-based RS.

Personalized search systems aim at providing a better search experience by returning the most relevant results, typically web pages (or resources), in response to a user's request. A clickthrough data (i.e., an event log of clicks on the search results after submitting a search query) can be used for this purpose as it contains information about users' actions and may provide valuable insights into search patterns. The essential part of these data is not just a web page that a user clicks on, but also a context, a query associated with every search request that carries a justification for the user's choice. The utility function in that case can be formulated as:

$$f_R : User \times Resource \times Query \rightarrow Relevance\ \ Score,$$

where *Resource* denotes a set of web pages and *Query* is a set of keywords that can be specified by users in order to emphasize their current interests or information needs. In the simplest case, a single query can consist of one or a few words (e.g., 'jaguar' or 'big cat'). More elaborate models could employ additional natural language processing tools in order to breakdown queries into a set of single keywords, e.g., a simple phrase 'what are the colors of the rainbow' could be transformed into a set {'rainbow,' 'color'} and further split into two separate queries, associated with the same (*user* and *resource*) pair.

CubeSVD

One of the earliest and at the same time very illustrative works where this formulation was explored with help of TF is CubeSVD.[45] The authors build a third-order tensor $\mathcal{Y} = [y_{ijk}]_{i,j,k=1}^{M,N,K}$. Values of the tensor represent the level of association (the relevance score) between the user i and the web-page j in presence of the query k:

$$\begin{cases} y_{ijk} > 0, & \text{if}(i,j,k) \in S, \\ y_{ijk} = 0, & \text{otherwise}, \end{cases}$$

where S is an observation history, e.g., a sequence of events described by the triplets (*user*, *resource*, and *query*). Note that authors in their work use simple queries without processing, e.g., 'big cat' is a single query term.

The association level can be expressed in various ways, the simplest one is to measure a co-occurrence frequency f, e.g., how many times a user has clicked on a specific page after submitting a certain query. In order to prevent an unfair bias towards the pages with high click rates, it can be restricted to have only values of 0 (no interactions) or 1 (at least one interaction). Or it can be rescaled with a logarithmic function:

$$f' = \log_2(1 + f/f_0),$$

where f' is a new normalized frequency and f_0 is, e.g., an Inverse Document Frequency (IDF) measure of a web page. Another scaling approach can also be used.

The authors proposed to model the data with a third-order TD (Eq. (13)) and in order to find it they applied the HOSVD. Similar to SVD (Eq. (3)), factors $U \in \mathbb{R}^{M \times r_1}, V \in \mathbb{R}^{N \times r_2}$ and $W \in \mathbb{R}^{K \times r_3}$ represent embedding of users, web pages, and queries vectors into a lower-dimensional latent factors space with dimensionalities r_1, r_2, and r_3 correspondingly. The core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ defines the form and the strength of multilinear relations between all three entities in the latent feature space. Once the decomposition is found, the relevance score for any (*user*, *resource*, and *query*) triplet can be recovered with Eq. (14).

With the introduction of new dimensions the data sparsity becomes even higher, which may lead to a numerical instabilities and general failure of the learning algorithm. In order to mitigate that problem, the authors propose several smoothing techniques: based on value imputation with small constant and based on the content similarity of web pages. They reported an improvement in the overall quality of the model after these modifications.

After applying the decomposition technique the reconstructed tensor $\mathcal{T}$ will contain new nonzero values denoting potential associations between users and web resources influenced by certain queries. The tensor values can be directly used to rank a list of the most relevant resources: the higher the value t_{ijk} is the higher the relevance of the page j to the user i within the query k.

This simple TF model does not contain a remedy for some of the typical RS problems such as cold start or real-time recommendations and is most likely to have issues with scalability. Nevertheless, this work is very illustrative and demonstrates the general concepts for building a tensor-based RS.

TOPHITS

As has been discussed in *Real-Time Recommendations* section, new entities can appear in the system dynamically and rapidly, which in the case of higher order models creates even more computational load, i.e., full recomputation of tensor decomposition quickly becomes infeasible and incremental techniques should be used instead. However, in some cases simply redefining the model might lower the complexity. As we mentioned in *Memory-Based Collaborative Filtering* section, a simple approach to reduce the model is to eliminate one of the entities with some sort of aggregation.

For example, instead of considering (*user*, *resource*, and *query*) triplets we could work with aggregated (*resource*, *resource*, and *query*) triplets, where every frontal slice $y_{:} : k$ of the tensor is simply an adjacency matrix of a resources browsed together under a specific query. Therefore users are no longer explicitly stored and their actions are recorded only in the form of a co-occurrence of resources they searched for.

An example of such a technique is TOPHITS model.[51,56] This analogy requires an extra explanation as the authors are not modeling users clicking behavior. The model is designed for web-link analysis of a static set of web pages referencing each other via hyperlinks. The data are collected by crawling those web pages and collecting not only links but also keywords associated with them. However, the crawler can be interpreted as a set of users browsing those sites by clicking on the hyperlinked keywords. This draws the connection between CubeSVD and TOPHITS model as the keywords can be interpreted as a short search queries in that case. And, as we stated earlier, users (or crawlers) can be eliminated from the model by constructing an adjacency matrix of linked resources.

The authors of TOPHITS model extend an adjacency matrix of interlinked web pages with the collected keyword information and build a so called adjacency tensor $T \in \mathbb{R}^{N \times N \times K}$, that encodes hubs, authorities and keywords. As has been mentioned, the keyword information is conceptually very similar to queries, hence it can be also modeled in a multirelational way. Instead of TD format the authors prefer to use CP in the form of Eq. (10) with $U, V \in \mathbb{R}^{N \times r}$ and $W \in \mathbb{R}^{K \times r}$ with ALS-based optimization.

The interpretation of this decomposition is different from the CubeSVD. As the authors demonstrate, the weights λ_k, $(1 \le k \le r)$ have a straightforward semantic meaning as they correspond to a set of r specific topics extracted from the overall web page collection. Accordingly, every triplet of vectors (u_k, v_k, w_k) represents a collection of hubs, authorities and keyword terms respectively, characterized by a topic k. The elements with higher values in these vectors provide the best-matching candidates under the selected topic, which allows a better grouping of web pages within every topic and provide means for a personalization.

For example, as the authors show, a personalized ranked list of authorities can be obtained with:

$$\mathbf{a}^* = V \Lambda W^T \mathbf{q}, \tag{21}$$

where $\Lambda = \mathrm{iag}(\lambda)$ is a diagonal matrix and $\mathbf{q}$ is a user-defined query vector of length K with elements $q_t = 1$ if term t belongs to the query and 0 otherwise, $t = \{1, \ldots, K\}$. Similarly, a personalized list of hubs can be built simply by substituting factor V with U in Eq. (21).

The interpretation of tensor values might appear very natural; however, there is an important note to keep in mind. Generally, the restored tensor values might turn both positive and negative. And in most applications the negative values have no meaningful explanation. The nonnegative TF (NTF)[57,58] can be employed to resolve that issue (see e.g., Ref 59 and also the connection of NTF to probabilistic factorization model under a specific conditions).[60]

Social Tagging

A remarkable amount of research is devoted to one specific domain, namely social tagging systems (STS), where predictions and recommendations are based on commonalities in social tagging behavior (also referred as collaborative tagging). A comprehensive overview of the general challenges and state-of-the-art RS methods can be found in Ref 61.

Tags carry a complementary semantic information that helps STS users to categorize and organize items of their choice. This brings an additional level of interpretation of the user–item interactions, as it exposes the motives behind the user preferences and explains the relevance of particular items. This observation suggests that tags play an important role in defining the relevance of (*user* and *item*) pairs, hence all the three entities should be modeled mutually in a multirelational way. The scoring function in that case can be defined as follows:

$$f_R : User \times Item \times Tag \rightarrow Relevance\ \ Score.$$

The triplets (*user*, *item*, and *tag*), coming from an observation history S, can be easily translated into a third-order tensor $\mathcal{Y} = [y_{ijk}]_{i,j,k=1}^{M,N,K}$, where M, N, K denote the number of users, items, and tags, respectively. Users are typically not allowed to assign the same tags to the same items more than once; hence the tensor values are strictly binary and defined as:

$$\begin{cases} y_{ijk} = 1, & \text{if}(i,j,k) \in S, \\ y_{ijk} = 0, & \text{otherwise.} \end{cases}$$

Unified Framework

As in the case of keywords and queries, tensor dimensionality reduction helps to uncover latent semantic structure of the ternary relations. The values of the reconstructed tensor $\mathcal{T}$ can be interpreted as the likeliness or weight of new links between users, items and tags. These links might be used for building recommendations in various ways: help users assign relevant tags for items,[62] find interesting new items,[63] or even find like-minded users.[64] The model that is built on top of all three possibilities is described in Ref 46. The authors perform a latent semantic analysis on the data with help of the HOSVD. Generally, the base model is similar to CubeSVD (see *Personalized Search and Resource Recommendations* section): items can be treated as resources and tags as queries.

The authors also face the same problem with sparsity. The tensor matricizations $Y_{(n)}$, $1 \le n \le 3$ within the HOSVD procedure produce highly sparse matrices which may prevent the algorithm from learning the accurate model. In order to overcome that problem they propose a smoothing technique based on a *kernel trick*.

In order to deal with the problem of real-time recommendations (see *Real-Time Recommendations* section) the authors adopt a well-known *folding-in method*[27] to a higher order case. The folding-in procedure helps to quickly embed a previously unseen entity into the latent features space without recomputing the whole model. For example, an update to a users feature matrix U can be obtained with:

$$\mathbf{u}_{new} = \mathbf{p} V_1 \Sigma_1^{-1},$$

where $\mathbf{p}$ is a new user information that corresponds to a *row* in the matricitized tensor $Y_{(1)}$; V_1 is an already computed (during HOSVD step) matrix of right singular vectors of $T_{(1)}$, and Σ_1 is a corresponding diagonal matrix of singular values; $\mathbf{u}_{new}$ is an *update row* which is appended to the latent factor matrix U. The resulting update to reconstructed tensor $\mathcal{T}$ is computed with (see Figure 3):

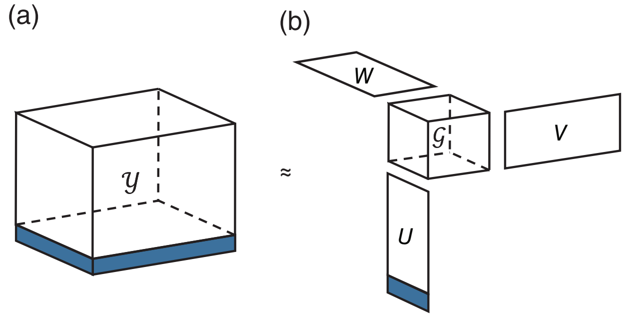

FIGURE 3 | Higher-order folding-in for Tucker decomposition. A slice with new user information in the original data (a) and a corresponding row update of the factor matrix in TD (b) are marked with solid color.

$$\mathcal{T}_{new} = [\mathcal{G} \times_2 V \times_3 W] \times_1 \begin{bmatrix} U \\ \mathbf{u}_{new} \end{bmatrix},$$

where the term within the left brackets of the right hand side does not contain any new values, e.g., does not require the full recomputation and can be prestored, which makes the update procedure much more efficient.

Nevertheless, this typically leads to a loss of orthogonality in factors and negatively impacts the accuracy of the model in the long run. This can be avoided with an *incremental SVD update*, which for the matrices with missing entries was initially proposed by Ref 26. As the authors demonstrate, it can be also adopted for tensors.

It should be noted, that this is not the only possible option for incremental updates. For example, a different incremental TD-based model with HOOI-based optimization is proposed in Ref 48 for a highly dynamic, evolving environment (not related to tag-based recommendations). The authors of this work use an extension of a two-dimensional incremental approach from.[65]

RTF and PITF

The models, overviewed so far, has a common '1/0' interpretation scheme for a missing values, i.e., all triplets $(i, j, k) \in S$ are assumed to be positive feedback and all others (missing) are negative feedback with zero relevance score. However, as the authors of ranking with TF model (RTF)[52] and more elaborate pairwise interaction TF (PITF)[54] model emphasize, all missing entries can be split into two groups: the true negative feedback and the unknown values. The true negatives correspond to those triplets of (i, j, k) where the user i has interacted with the item j and has assigned tags different from the tag k. More formally, if P_S is a set of all posts that correspond to all observed (*user* and *item*) interactions, than true negative feedback within any interaction is defined as:

$$Y_{ij}^{-} := \{k | (i,j) \in P_S \wedge (i,j,k) \notin S\};$$

trivially, true positive feedback is:

$$Y_{ij}^{+} := \{k | (i,j) \in P_S \wedge (i,j,k) \in S\}.$$

All other entries are unknowns and are to be uncovered by the model.

Furthermore, both RTF and PITF models do not require any specific values to be imposed on either known or unknown entries. Instead they only impose pairwise ranking constraints on the reconstructed tensor values:

$$t_{ijk_1} > t_{ijk_2} \Leftrightarrow (i,j,k_1) \in Y_{ij}^{+} \wedge (i,j,k_2) \in Y_{ij}^{-}.$$

These post-based ranking constraints become the essential part of an optimization procedure. The RTF model uses the Tucker format; however, it aims at directly maximizing AUC measure, which, according to the authors, takes the following form:

$$AUC(\theta,i,j) := \frac{1}{\left|Y_{ij}^{+}\right|\left|Y_{ij}^{-}\right|} \sum_{k^+ \in Y_{ij}^{+}} \sum_{k^- \in Y_{ij}^{-}} \sigma\left(t_{ijk^+} - t_{ijk^-}\right),$$

where θ are the parameters of the TD model (as defined in Eq. (18)) and $\sigma(x)$ is a sigmoid function, introduced to make the term differentiable:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{22}$$

As we are interested in maximizing AUC and due to Eq. (16), the loss function takes the form:

$$L(\mathcal{Y}, \mathcal{T}) = -\sum_{(i,j) \in P_S} AUC(\theta,i,j).$$

The regularization term of the model is defined by Eq. (20).

The authors adopt an SGD algorithm for solving the optimization task. However, as they state, directly optimizing the AUC objective is computationally infeasible. Instead, they exploit a smart trick of recombining and reusing precomputed summation terms within the objective and its derivatives. They use this trick for both tasks of learning and building recommendations.

The PITF model is built on top of ideas from RTF model. It adopts Bayesian Personalized Ranking (BPR) technique proposed for MF case in Ref 66 to the ranking approach. The tags rankings for every observed post (i, j) are not deterministically learned like in RTF model but instead are derived from the observations by optimizing the maximum *a posteriori* estimation. This leads to a similar to RTF optimization objective with similar regularization (excluding the tensor core term which is not present in CP) and slightly different loss function:

$$L(\mathcal{Y}, \mathcal{T}) = -\sum_{(i,j,k_1,k_2) \in D_S} \ln \sigma\left(t_{ijk_1} - t_{ijk_2}\right),$$

where the same notation as in RTF is used; σ is a sigmoid function from Eq. (22) and D_S is a training data, i.e., a set of quadruples:

$$D_S = \{(i,j,k_1,k_2) | (i,j,k_1) \in S \wedge (i,j,k_2) \notin S\}. \tag{23}$$

An important difference of PITF from RTF is that the complexity of multilinear relations is significantly reduced by leaving only pairwise interactions between all entities. From the mathematical viewpoint it can be considered as a CP model with a special form of partially fixed factor matrices (cf. Eq. (12)):

$$t_{ijk} = \sum_{\alpha} u_{i\alpha} w_{k\alpha}^{U} + \sum_{\alpha} v_{j\alpha} w_{k\alpha}^{V} + \sum_{\alpha} u_{i\alpha} v_{j\alpha}, \tag{24}$$

where $w_{k\alpha}^{U}$ and $w_{k\alpha}^{V}$ are the parts of the same matrix W responsible for tags relation to users and items respectively; $u_{i\alpha}$ and $v_{j\alpha}$ are interactional parts of U and V.

The authors emphasize, that the user–item interaction term does not contribute to the BPR-based ranking optimization scheme which yields even more simple equation that becomes an essential part of the PITF model:

$$t_{ijk} = \sum_{\alpha} u_{i\alpha} w_{k\alpha}^{U} + \sum_{\alpha} v_{j\alpha} w_{k\alpha}^{V}. \tag{25}$$

Another computational trick that helps to train the model even faster without sacrificing the quality is random sampling within the SGD routine. All the quadruples in D_S corresponding to a post (i, j) are highly overlapped with respect to the tags associated with them. Therefore, learning with some randomly sampled quadruples is likely to have a positive effect on learning the remaining ones.

In order to verify the correctness and effectiveness of such simplifications, the authors conduct experiments with both BPR-tuned TD and CP and demonstrate that PITF algorithm achieves close or even better quality of recommendations while learning features faster than the other two TF methods.

Despite its computational effectiveness, the original PITF model is lacking the support for the real-time recommendation scenarios, where rebuilding the full model for each new user, item or tag could be prohibitive. The authors of Ref 67 overcome this limitation by introducing the folding-in procedure compatible with the PITF model and demonstrate its ability to provide high recommendations quality. Worth noting here, that a number of variations of the folding-in technique are available for different TF methods, see Ref 68. The idea of modeling higher order relations in a joint pairwise manner similar to Eq. (25) has been explored in various application domains and is implemented in various settings, either straightforwardly or as a part of a more elaborate RS model.[69–72] There are several generalized models,[23,33,73] that also use this idea. They are covered in more details in *CARTD* and *GFF* sections of this study.

Improving the Prediction Quality

As has been already mentioned in *Unified Framework* section, high data sparsity typically leads to a less accurate predictive models. This problem is common across various RS domains. Another problem, specific to STS, is tag ambiguity and redundancy. The following are the examples of some of the most common techniques, developed to deal with these problems.

The authors of CubeRec[74] propose a clustering-based separation mechanism. This mechanism builds clusters of triplets (*user*, *item*, and *tag*) based on the proximity of tags derived from the item–tag matrix. With this clustering, some of the items and tags can belong to several clusters at the same time, according to their meaning. After that, the initial problem is split into a number of subproblems (corresponding to clusters) of a smaller size and hence, with a more dense data. Every subproblem is then factorized with the HOSVD similar to Ref 45, and the resulting model is constructed as a combination of all the smaller TF models.

The authors of the clustering-based TD model (ClustHOSVD)[47] also employ clustering approach. However, instead of splitting the problem, they replace tags by tag clusters and apply the HOOI method (which is named AlsHOSVD by the authors) directly to the modified data consisting of (*user*, *item*, and *tag cluster*) triplets. They also demonstrate the effect of different clustering techniques on the quality of RS.

As can be seen, many models benefit from clustering either prior to or after the factorization step. This suggests that it can also be beneficial to perform simultaneous clustering and factorization. This idea is explored by the authors of Ref 75, where they demonstrate the effectiveness of such an approach.

A further improvement can be achieved with hybrid models (see *Model-Based Collaborative Filtering* section), that exploit a content information and incorporate it into a tensor-based CF model. It should be noted, however, that there is no 'single-bullet' approach, suitable for all kinds of problems, as it highly depends on the type of data used as a source of content information.

The authors of Ref 76 exploit acoustic features for music recommendations in a tag-based environment. The features, extracted with specific audio-processing techniques, are used to measure the similarity between different music samples. The authors make an assumption that similarly sounding music is likely to have similar tags, which allows to propagate tags to the music that was not tagged yet. With this assumption the data is augmented with new triplets of (*user*, *item*, and *tag*), which leads to a more dense data and results in a better predictive quality of the HOSVD model.

The TF and tag clustering (TFC) model[44] combines both content exploitation and tag clustering techniques. The authors focus on the image recommendations problem, thus they use an image processing techniques in order to find items' similarities and propagate highly relevant tags. Once the tag propagation is completed, the authors find tag clusters (naming them topics) and build new association triplets (*user*, *item*, and *topic*), which are further factorized with the HOSVD.

As a last remark in this section, the idea of model splitting, proposed in the CubeRec model, was also explored in a more general setup in Ref 77. The authors consider a multiple context environment, where user–item interactions may depend on various contexts such as location, time, activity, and so on. This is generally modeled with an Nth order tensor, where $N > 3$. Instead of dealing with higher number of dimensions and greater sparsity, the authors propose to build a separate model for every context type, which transforms the initial problem into a collection of smaller problems of order 3. Then all the resulting TF models are combined with specific weights (based on the context influence measure proposed by the authors) and can be used to produce recommendations. However, despite the ability to better handle the sparsity issue, the model may loose some valuable information about the relations between different types of context. More general methods for multicontext problems are covered in *General Context-Aware Models* section.

Temporal Models

User consumption patterns may change in time. For example, the interest of TV users may correlate not only with a topic of a TV program, but also with a specific time of the day. In retail user preferences may vary depending on the season. Temporal models are designed to learn those time-evolving patterns in data by taking the time aspect into account, which can be formalized with the following way scoring function:

$$f_R : User \times Item \times Time \rightarrow Relevance\ \ Score.$$

Even though the general problem statement looks already familiar, when working with the *Time* domain one should mind the difference between the evolving and periodic (e.g., seasonal) events which may require a special treatment.

BPTF

One of the models that exploits periodicity of events is the Bayesian Probabilistic TF (BPTF).[78] It uses seasonality to reveal trends in retail data and predict the orders that will arrive in the ongoing season based on the season's start and previous purchasing history. The key feature of the model is the ability to produce forecasts on the sales of the new products that were not present in previous seasons. The model captures dynamic changes in both product designs and customers' preferences solely from the history of transactions and does not require any kind of an expert knowledge.

The authors develop a probabilistic latent factors model by introducing priors on the parameters; i.e., the latent feature vectors are allowed to vary and the variance of relevance scores is assumed to follow a Gaussian distribution:

$$t_{ijk}|U, V, W \sim \mathcal{N}\left(< U_{i:}, V_{j:}, W_{k:} >, \ \gamma^{-1}\right),$$

where γ is an observations precision and $< U_{i:}, V_{j:}, W_{k:} >$ denotes a right hand side of Eq. (12). Note, that in the original work the authors use a transposed version of the factor matrices, e.g., any column of the factor U in their work represents a single user, the same holds for two other factors.

In order to prevent the overfitting the authors also impose prior distributions on U and V:

$$U_{i:} \sim \mathcal{N}\left(0, \ \sigma_U^2 \ I\right),$$
$$V_{j:} \sim \mathcal{N}\left(0, \ \sigma_V^2 \ I\right).$$

Furthermore, the formulation for the time factor W takes into account its evolving nature and implies smooth changes in time:

$$W_{k:} \sim \mathcal{N}\left(W_{k-1:}, \ \sigma_{dW}^2 \ I\right),$$
$$W_{0:} \sim \mathcal{N}\left(\mu_W, \ \sigma_0^2 \ I\right).$$

The time factor W rescales the user–item relevance score with respect to the time-evolving trends and the probabilistic formulation helps to account for the users who do not follow those trends.

The authors show that maximizing the log-posterior distribution $\log p(U, V, W, W_{0,:}|\mathcal{Y})$ with respect to U, V, W and $W_{0\,:}$ is equivalent to an optimization task with the weighted square loss function:

$$L(\mathcal{Y}, \mathcal{T}) = \sum_{(i,j,k) \in S} \left(y_{ijk} - t_{ijk}\right)^2, \qquad (26)$$

and a bit more complex regularization term:

$$\Omega(\theta) = \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_V}{2}||V||_F^2 + \frac{\lambda_{dT}}{2}\sum_{k=1}^{K}||W_{k:} - W_{k-1:}||^2 + \frac{\lambda_0}{2}||W_{0:} - \mu_W||^2,$$

where $\lambda_U = (\alpha\sigma_U)^{-1}$, $\lambda_V = (\alpha\sigma_V)^{-1}$, $\lambda_{dW} = (\alpha\sigma_{dW})^{-1}$, $\lambda_0 = (\alpha\sigma_0)^{-1}$ and the last two terms are due to a dynamic problem formulation. The number of parameters of this model makes the task of optimization almost infeasible. However, the authors come up with an elaborate Markov chain Monte Carlo (MCMC) integration approach that makes the model almost parameter-free and also scales well.

TCC

The authors of TF-based subspace clustering and preferences consolidation model (TCC)[79] exploit the periodicity in usage patterns of the IPTV users in order to, at first, identify them and, secondly, provide with more relevant recommendations, even if those users share the same IPTV account (e.g., across all family members). This gives a slightly different definition of a utility function:

$$f_R : Account \times Item \times Time \rightarrow Relevance\ \ Score,$$

where *Account* is the domain of all registered accounts and the number of accounts is not greater than the number of users, i.e., $|Account| \leq |User|$. Initial tensor $\mathcal{Y}$ is built from the triplets (*account*, *item*, and *time*) and its values are just the play counts.

In order to be able to find a correct mapping of the real users to the known accounts, the authors introduce a concept of a *virtual user*. Within the model the real user is assumed to be a composition of particular virtual users u_{ak} which express the specific user's preferences tied to a certain time periods, for example:

$$u_{ak} := \{(a, p_k) \mid a \in A,\ p_k \in P,\ p_k \neq \emptyset\},$$

where a is an account from the set of all accounts A and p_k is a subperiod from the set of all nonoverlapping time periods P.

As the authors state, manually splitting the time data into the time slots p_k does not fit the data well and they propose to find those subperiods from the model. They first solve the SGD-based optimization task (Eqs (18) and (16)) for the TD with the same weighted squared loss function as in Eq. (26) and regularization term as in Eq. (19) (with $\lambda_G = \lambda_U = \lambda_V = \lambda_W = \frac{1}{2}\lambda$). Once the model factors are found, the subperiods p_k can be obtained by clustering the time feature vectors:

$$P \leftarrow \text{k-Means clustering of the rows of } W.$$

Then the consolidation of virtual users into the real ones can be performed in two steps. At first, a binary similarity measure is computed between different pairs of virtual users (u_{ak}, $u_{ak'}$) corresponding to the same account a. The second step is to combine similar virtual users so that every real user is represented as a set of virtual ones. This is performed with help of graph-based techniques. Once the real users are identified, recommendations can be produced with a user-based kNN approach. As the authors demonstrate, the proposed method not only provides a tool for user identification, but also outperforms standard kNN and TF-based methods applied without any prior clustering.

General Context-Aware Models

In previous sections, we have discussed TF methods targeted at specific classes of problems: keyword- or tag-based recommendations, temporal models. They all have one thing in common—the use of a third entity leading to a higher level of granularity and better predictive capabilities of a model. This leads to an idea of generalization of such an approach that is suitable for any model formulated in the form of Eq. (2).

Multiverse

One of the first attempts towards this generalization is the Multiverse model.[32] The authors define context as any set of variables that influence users' preferences and propose to model it by the Nth order TD with $N - 2$ contextual dimensions:

$$\mathcal{T} = [\![\mathcal{G}; U, V, W_1, W_2, \ldots, W_{N-2}]\!],$$

where factors W_i, $i = \{1, \ldots, N-2\}$ represent a corresponding embedding of every contextual variable into a reduced latent space and all factors including U and V are not restricted to be orthogonal. As the authors state, the model is suitable for any contextual variables over a finite categorical domain. It should be noted, that the main focus of the work is systems with an explicit feedback and the model is optimized for the error-based metrics, which does not guarantee an optimal items ranking as has been discussed in *Model Evaluation* section.

Following the general form of an optimization objective stated in Eq. (18), the authors use the weighted loss function:

$$L(\mathcal{T}, \mathcal{Y}) = \frac{1}{\|\mathcal{G}\|_1} \sum_{(i,j,k) \in S} l(t_{ijk},\ y_{ijk}),$$

where $l(t_{ijk},\ y_{ijk})$ is a pointwise loss function, that can be based on l_2, l_1, or other types of distance measure. The example is provided for the third-order case; however, it can be easily generalized to higher orders. The authors also use the same form of the regularization term as in Eq. (19), as it enables trivial optimization procedure.

In order to fight against the growing complexity for the higher order cases they propose a modification of the SGD algorithm. Within a single optimization step, the updates are performed on every row of the latent factors independently. For example, an update for ith row of U:

$$U_{i:} \leftarrow U_{i:} - \eta\lambda_U U_{i:} - \eta\partial_{t_{ijk}} l(t_{ijk},\ y_{ijk})$$

is independent on all other factors and thus all the updates can be performed in parallel. The parameter η defines the model's learning step.

In addition to the general results on the real dataset, this work features a comprehensive experimentation on the semi-synthetic data that shows the impact of contextual information on the RS models performance. It demonstrates that high context influence leads to better quality of the selected context-aware methods, among which the proposed TF approach gives the best results, while a context-unaware method's quality significantly degrades.

TFMAP

Similar to the previously discussed PITF model, the TF for MAP optimization model (TFMAP)[53] also targets optimal ranking, however it exploits the MAP metric instead of AUC. The model is designed for implicit feedback systems, which means that the original tensor $\mathcal{Y}$ is binary with nonzero elements reflecting the fact that the interaction has occurred:

$$y_{ijk} = \begin{cases} 1, & \text{if}(i,j,k) \in S, \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

The optimization objective is drawn from the MAP definition:

$$MAP = \frac{1}{MK} \sum_{i=1}^{M} \sum_{k=1}^{K} \frac{\sum_{j=1}^{N} \frac{y_{ijk}}{r_{ijk}} \sum_{j'=1}^{N} y_{ij'k} \mathbb{I}\left(r_{ij'k} \le r_{ijk}\right)}{\sum_{j=1}^{N} y_{ijk}},$$

where r_{ijk} denotes the rank of the item j in the items list of the user i under the context type k and $\mathbb{I}(\cdot)$ is an indicator function, which is equal to 1 if the condition is satisfied and 0 otherwise, both depend on the reconstructed values of $\mathcal{T}$. In order to make the metric smooth and differentiable, the authors propose two approximations:

$$\frac{1}{r_{ijk}} \approx \sigma\left(t_{ijk}\right),$$

$$\mathbb{I}\left(r_{ij'k} \le r_{ijk}\right) \approx \sigma\left(t_{ij'k} - t_{ijk}\right),$$

where t_{ijk} is calculated with Eq. (12) (which makes the model a CP-based) and σ is a sigmoid function defined by Eq. (22). Notably, $t_{ij'k} - t_{ijk} = \langle U_{i:}, V_{j':} - V_{j:}, W_{k:} \rangle$, where we use the same notation as in BPTF model, see *BPTF* section.

The model also follows the standard optimization formulation stated in Eq. (18), where the loss function is just a negative MAP gain, i.e., $L(\mathcal{T}, \mathcal{Y}) = -MAP$, and the regularization has the form of Eq. (20).

Note, that MAP optimization also has a weighted form due to Eq. (27); however, the computation complexity would still be prohibitively high due to its complex structure. In order to mitigate that, the authors propose the fast learning algorithm: for each (*user* and *context*) pair only a limited set of a representative items (a buffer) is considered, which in turn, allows to control the computational complexity. They also provide an efficient algorithm of sampling the 'right' items and constructing the buffer, which does not harm the overall quality of the model.

CARTD

The Context-Aware Recommendation TD (CARTD) model[33,73] provides a generalized framework for an arbitrary number of contexts and also targets an optimal ranking instead of a rating prediction. Under the hood, the model extends the BPR-based ranking approach used in the PITF model to the higher order cases.

The authors introduce a unified notion of an entity. A formal task is to find the list of the most relevant entities within a given contextual situation. Remarkably, all the information, that is used to make recommendations more accurate and relevant, is defined as a context. In that sense, not only information like tag, time, location, user attributes, and so on, is considered to be a context, even users themselves might be defined as a context of an item. This gives a more universal formulation for the recommendations task:

$$f_R : Entity \times Context_1 \times \ldots \times Context_n \rightarrow Relevance\ \ Score. \tag{28}$$

As an illustration to that, a quadruple (*user*, *item*, *time*, and *location*) maps to (*context*$_1$, *entity*, *context*$_2$, and *context*$_3$). Obviously, the definition of the entity depends on the task. For example, in case of social interactions prediction with (*user*, *user*, and *attribute*) triplets, the main entity as well as one of the context variables will be a user.

The observation data in a typical case of user–item interactions can be encoded similar to Eq. (23):

$$D_S := \{(e,f,c_1,\ldots,c_n) \mid (e,c_1,\ldots,c_n) \in S \wedge (f,c_1,\ldots,c_n) \notin S\},$$

where e and f are the entities (i.e., items) and c_i, $i = \{1, \ldots, n\}$ denotes a context type (includes users). As the authors emphasize, this leads to a huge sparsity problem, and instead they propose to relax conditions and instead build the following set for learning the model:

$$D_A := \{(e,f,c_1,\ldots,c_n) \mid \forall c_i :_{c_i}(e) >_{c_i}(f)\},$$

where $\#_{c_i}(\cdot)$ indicates the number of occurrences of an entity within the context $\#c_i$. The rule $\#_{c_i}(e) >_{c_i}(f)$ denotes the prevalence of the entity e over the entity f with respect to all possible contexts.

The optimization objective will also look similar to the one used in the PITF model with the loss function defined as:

$$L(\mathcal{T},\mathcal{Y}) = -\sum_{(e,f,c_1,\ldots,c_n)\in D_A} \ln\sigma\left(t_{\{c\},e} - t_{\{c\},f}\right),$$

where $\{c\}$ denotes a set of all context variables c_i and the tensor values t_{ijk} are calculated with help of the reduced CP model with the pairwise only interactions, similar to Eq. (25):

$$t_{\{c\},e} = \sum_{i=1}^{n} v_e^{E,C_i} v_{c_i}^{C_i,E},$$

where v_e^{E,C_i} and $v_{c_i}^{C_i,E}$ are the elements at the cross section of the eth row and the ith column of the factor matrices V^{E,C_i} and $V^{C_i,E}$ respectively. As in the previous cases, the regularization term $\Omega(\theta)$ have similar to Eq. (20) form, which includes all the factors from θ:

$$\theta = \left\{V^{E,C_1}, V^{C_1,E},\ldots,V^{E,C_n}, V^{C_n,E}\right\}.$$

iTALS

As has been mentioned in the introduction (see *Implicit Feedback* section), an implicit feedback does not always correlate with the actual user preferences, thus a simple binary scheme (as in Eq. (27)) may not be accurate enough. For this reason, the authors of the iTALS model (ALS-based implicit feedback recommender algorithm)[80] propose to use the confidence-based interpretation of an implicit feedback introduced in Ref 17 and adopt it for the higher order case.

They introduce the dense tensor $\mathcal{W}$ that assigns nonzero weights for both observed and unobserved interactions. For the nth order tensor, it has the following form:

$$\begin{cases} w_{i_1,\ldots,i_n} = \alpha\cdot\#(i_1,\ldots,i_n), & \text{if}(i_1,\ldots,i_n)\in S,\\ w_{i_1,\ldots,i_n} = 1, & \text{otherwise,}\end{cases} \tag{29}$$

where $\#\,(i_1, \ldots, i_n)$ is the number of occurrences of the tuple $(i_1, \ldots, i_n)$ (e.g., the combination of the user i_1 and the item i_2 interacted within the set of contexts $i_3, \ldots, i_n$) in the observation history; α is set empirically and $\alpha \cdot \#\,(i_1, \ldots, i_n) > 1$ which means that the observed events provide more confidence in the user preferences than the unobserved ones.

The loss function will then take the form:

$$L(\mathcal{T},\mathcal{Y}) = \sum_{i_1,\ldots,i_n} w_{i_1,\ldots,i_n}\left(t_{i_1,\ldots,i_n} - y_{i_1,\ldots,i_n}\right)^2,$$

where weights $w_{i_1,\ldots,i_n}$ are defined by Eq. (29), $y_{i_1,\ldots,i_n}$ are the values of a binary feedback tensor of order n, defined similar to Eq. (27), and $t_{i_1,\ldots,i_n}$ are the values of the reconstructed tensor.

The model uses CP with an ALS-based optimization procedure and a standard regularization similar to Eq. (20). The latent feature vectors are encoded in the rows of the factor matrices, not the columns, i.e., following the authors' notation, we should rewrite Eq. (11) as:

$$\mathcal{T} = [\![M_1^T,\ldots,M_n^T]\!],$$

where M_i $(1 \le i \le n)$ are transposed factors of the CP decomposition.

The authors show, how an efficient computation over the dense tensor can be achieved with the same tricks that are used in Ref 17 for the matrix case. The model also has a number of modifications:[81] based on the conjugate gradient approach (iTALS-CG) and the coordinates descent approach (iTALS-CD) where an additional features compression is achieved by the Cholesky decomposition. This makes the iTALS-CD model to learn even faster than MF methods. While performing on approximately the same level of accuracy as the state-of-the-art Factorization Machines (FM) method,[82] it is capable of learning more complex latent relations structure. Another modification is the pairwise 'PITF-like' reduction model, named iTALSx.[70]

GFF

The General Factorization Framework (GFF)[23] further develops the main ideas of the family of iTALS models. Within the GFF model different CP-based factorization models (also called a preference models) are combined in order to capture the intrinsic relations between users and items influenced by an arbitrary number of contexts. As in many other works the authors of GFF model fix the broad definition of the context as an entity, which 'value is not determined solely by the user or the item,' i.e., not a content information (see *Incorporating Context Information* section).

The model can be better explained with the example. Let us consider the problem of learning the scoring function as follows:

$$f_R : U \times I \times S \times Q \rightarrow Relevance\ Score, \quad (30)$$

where U and I are the domains of *users* and *items*, respectively; S stands for *season* and denotes the periodicity of the events (see *Temporal Models* section); Q describes the sequential consumption patterns, e.g., what are the previous items that were also consumed with the current one (see Ref 80 for broader set of examples). Let us also define the pairwise interactions between users and items as *UI* (standard CF model), between items and seasons as *IS* and so forth. Using the same notation, we can also define multirelational interactions, such as *UIS* for a three-way user–item–season interactions or *UISQ* for the four-way interactions between all four types of entities.

In total, there could be 2047 different combinations of interactions, yet not all of them are feasible in terms RS model, as not all of them may contribute to the preference model.

As the result, GFF generates a very flexible multirelational model that allows to pick the most appropriate scheme of interactions, that does not explode the complexity of the model and meanwhile achieves a high quality of recommendations. Based on the experiments, the authors conclude: 'leaving out useless interactions results in more accurate models.'

We have reviewed so far a diverse set of tensor-based recommendation techniques. Clearly, tensors help represent and model complex environments in a convenient and congruent way, suitable for various problem formulations. Nevertheless, as we have already stated earlier, the most common practical task for RS is to build a ranked list of recommendations (a top-n recommendations task). In this regard, we summarize related features of some of the most illustrative in our opinion methods in Table 2. We also take into account a support for real-time scenarios in dynamic environments.

Other Models

Unfortunately, it is almost impossible to review all available TF models from various domains. The flexibility that comes with the tensor-based formulation provides means for limitless combinations of various RS settings and models. Here we briefly describe some of them that were not referenced yet, but have an interesting application and/or implementation.

Social Interactions

The authors of Ref 83 focus on recommending new connections for users in specific communities like online dating or professional networks They emphasize that typically there are two types of groups of people (e.g., employee and employer) in job seeking networks. In order to account for that and avoid unnecessary recommendations within the same group (e.g., employer–employer) they split the problem into two parallel subproblems corresponding to each individual group and model it with the CP. The final result is than aggregated from both subproblems by selecting

TABLE 2 | Comparison of Popular Tensor Factorization Methods

Name	Type	Algorithm	Domain	Entities	Optimization	Ranking Prediction	Online
TOPHITS,[56] 2005	CP	ALS	Link prediction	Resources, keyword	Pointwise	Yes	No
CubeSVD,[45] 2005	TD	HOSVD	Personalized search	User, resource, query	Pointwise	Yes	No
RTF,[52] 2009	TD	SGD	Folksonomy	User, item, tag	Pairwise	Yes	No
BPTF,[78] 2010	CP	MCMC	Temporal	User, item, time	Pointwise	No	No
Multiverse,[32] 2010	TD	SGD	Context-awareness	User, item, contexts	Pointwise	No	No
PITF,[54] 2010	CP[1]	SGD	Folksonomy	User, item, tag	Pairwise	Yes	No
TagTR,[46] 2010	TD	HOSVD	Folksonomy	User, item, tag	Pointwise	Yes	Yes
TFMAP,[53] 2012	CP	SGD	Context-awareness	User, item, context	Listwise	Yes	No
CARTD,[33] 2012	CP[1]	SGD	Context-awareness	Item, contexts	Pairwise	Yes	No
ClustHOSVD,[47] 2015	TD	HOOI	Folksonomy	User, item, tag	Pointwise	Yes	No
GFF,[23] 2015	CP[1]	ALS	Context-awareness	User, item, contexts	Pointwise	Yes	No

ALS, alternating least squares; CP, Candecomp/Parafac; HOOI, Higher-Order Orthogonal Iteration; HOSVD, Higher-Order Singular Value Decomposition; MCMC, Markov chain Monte Carlo; SGD, stochastic gradient descent; TD, Tensor Decomposition.
The *Ranking prediction* column shows whether a method is evaluated against ranking metrics. The *Online* column denotes a support for real-time recommendations for new users (e.g., folding-in).
[1] Method uses pairwise reduction concept, initially introduced in PITF.

only those predicted links (i.e., recommendations) which are present in both groups.

The TOPHITS approach, described in *TOPHITS* section, is shown to be applicable for the authorities ranking task in the Twitter community.[84] This technique can be potentially used for improving the followee recommendations for twitter users.

Social Tagging

A few works for image tagging[85,86] use a interesting representation of data, initially proposed in Ref 87. Users and images, uploaded by them in social network, are encoded together into a single long vector. These vectors are used to build a set of adjacency matrices that are made with respect to certain conditions and then stacked together in a tensor. With this approach, every frontal slice of the tensor describes different kinds of relations: friendship relations between users, user–image connections, tag relations for both users and items, and so on.

Temporal Models

The authors of Ref 88 add a so called *social regularization*, introduced in Ref 89 into a standard optimization routine. The idea behind this modification is to use not only a 'wisdom of crowds' like in standard CF approach, but also utilize information about social relationships (i.e., friendship) of the user in order to bring more trust into the recommendations an improve the overall accuracy.

The work[90] combines both social tagging and temporal models. The authors build a fourth-order tensor from (*user*, *item*, *tag*, and *time*) quadruples and decompose it with the HOSVD. In order to estimate relevance scores and recommend new items for users, they first summarize values, corresponding to the observations, over the third (tag) mode.

An interesting hybrid approach for modeling user preferences dynamics is proposed in Ref 91. The authors build a tensor from (*user*, *item*, and *time-period*) triplets and combine it with an auxiliary content information (user attributes) with help of a *coupled tensor–matrix factorization method*.[92,93] The idea of coupled TF–MF provides an additional level of flexibility in model construction and is used in various RS domains with complex setup.[94,95]

Multicriteria Ratings Systems

The authors of Ref 96 explore the rich sentiment information in a product reviews. They extract opinions from text and craft a multiaspect (or multicriteria, see Ref 97) ratings system on top of it. These data are used to build a third order tensor in the form (*user*, *item*, and *aspect*), with tensor values denoting the ratings within each aspect (including the explicit ratings). A CP-based factorization model is used to reconstruct missing values and predict unknown ratings more accurately.

A similar idea of multicriteria ratings model was also used as a part of a sophisticated model in Ref 98. However, the authors did not have to do any text analysis as the aspect data was populated by users themselves and provided within the dataset. They also applied the HOSVD method instead of the CP.

Mobility and Geolocation

Modern social networks allow to share not only a content, such as images or videos, but also link that content to specific locations using the Global Positioning System (GPS) services. With the broad access of mobile devices to the internet, this provides rich information about user interests and behavior and allows building highly personalized context-aware services and applications. For example, the authors of Refs 99,100 model location-based user activities with a third order tensor (*user*, *activity*, and *location*) for providing locations and activities recommendations. The authors of Ref 101 use the tensor for the personalized activities rating prediction. These works use the Tucker tensor format and apply the HOSVD for its reconstruction.

Cross-Domain Recommendations

Another interesting direction is combining a cross-domain knowledge, e.g., user consumption patterns of books, movies, music, and so on, in order to improve recommendations quality. Knowledge about the patterns from one domain may help to build more accurate predictions in another (this is a so called *knowledge transfer* learning). Moreover, modeling these cross-domain relations mutually may also help to achieve a higher recommendations quality across all domains. An interesting challenge in these tasks is a varying number of items in different domains, which requires a special treatment. A few notable and quite different techniques of the tensor-based knowledge transfer learning are proposed in Refs 102 and 103.

Special Factorization Methods

In the theory of matrix approximations, there is well-known pseudo-skeleton decomposition method,[104] that allows to use only a small sample of the original matrix elements in order to find an approximate MF within the desired accuracy. This result is shown to be generalizable to a higher order case,[105,106] and, remarkably, is especially suitable for sparse data. The main benefit of such a sampling approach is the reduced factorization complexity in terms of both the number of operations and the number of elements required for computation, which is especially advantageous in case of tensor-based models. A special case of such a class of TF algorithms is used in the TensorCUR model[107] for product recommendations.

CONCLUSION

In this survey, we have attempted to overview a broad range of tensor-based methods used in RS to date. As we have seen, these methods provide powerful set of tools for merging various types of additional information that increases flexibility, customizability, and quality of recommendation models. Tensorization enables creative and nontrivial setups, going far beyond standard user–item paradigm, and finds its applications in various domains. Tensor-based models can also be used as a part of more elaborate systems, providing compressed latent representations as an input for other well-developed techniques.

One of the main concerns for the higher order models is inevitable growth of computational complexity with increasing number of dimensions. Even for mid-sized production systems, that have to deal with highly dynamic environments, this might have negative implications, such as inability to produce recommendations for new users instantly, in a timely manner. This type of issues can be firmly addressed with incremental update and higher order folding-in techniques. The former allow to update the entire model, while performing computations only on new data. The latter allows to calculate recommendations in cases when new data is already present in the system but was not yet included into the model.

Despite the encouraging results, there is an issue related to the applicability of CP and TD decompositions. When the number of dimensions becomes much higher than 3, application of TD-based methods becomes infeasible due to explosion of storage requirements. However, CP is generally ill-posed which may potentially lead to numerical instabilities. A possible cure for this problem is to use TT/HT decomposition. In our opinion, this is a promising direction for further investigations.

ACKNOWLEDGMENTS

This work was supported by Russian Science Foundation grant 14–1100659. The authors would like to thank Maxim Rakhuba and Alexander Fonarev for their help for improving the manuscript, and also Michael Thess for insightful conversations.

REFERENCES

1. Adomavicius G, Sankaranarayanan R, Sen S, Tuzhilin A. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans Inf Syst* 2005, 23:103–145.
2. Brusilovsky P. Social information access: the other side of the social web. In: *SOFSEM 2008 Theory and Practice of Computer Science*. Berlin Heidelberg: Springer-Verlag; 2008; 5–22.
3. Ekstrand MD, Riedl JT, Konstan JA. Collaborative filtering recommender systems. *Found Trends Hum-Comput Interact* 2011, 4:81–173.
4. Herlocker JL, Konstan JA, Terveen LG, Riedl JT. Evaluating collaborative filtering recommender systems. *ACM Trans Inf Syst* 2004, 22:5–53.
5. Knijnenburg BP, Willemsen MC. Evaluating recommender systems with user experiments. In: Ricci F, Rokach L, Shapira B, eds. *Recommender Systems Handbook*. New York: Springer; 2015, 309–352.
6. Bobadilla J, Ortega F, Hernando A, Gutiérrez A. Recommender systems survey. *Knowl-Based Syst* 2013, 46:109–132.
7. Ricci F, Rokach L, Shapira B. *Recommender Systems: Introduction and Challenges*. New York: Springer; 2015, 1–34.
8. Shi Y, Larson M, Hanjalic A. Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. *ACM Comput Surv* 2014, 47:1–45.
9. Lops P, Gemmis MD, Semeraro G. Content-based recommender systems: state of the art and trends. In: Ricci F, Rokach L, Shapira B, Kantor PB, eds. *Recommender Systems Handbook*. New York: Springer; 2011, 73–105.
10. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state of the art and possible extensions. *IEEE Trans Knowl Data Eng* 2005, 17:734–749.

11. Hastie T, Tibshirani R, Friedman J, Franklin J. The elements of statistical learning: data mining, inference and prediction. *Math Intell* 2005, 27:83–85.
12. Schafer JB, Frankowski D, Herlocker J, Sen S. *The Adaptive Web: Methods and Strategies of Web Personalization*. Collaborat. Berlin Heidelberg: Springer; 2007, 291–324.
13. Parra D, Brusilovsky P. Collaborative filtering for social tagging systems: an experiment with CiteULike. In: *Proceedings of the third ACM conference on Recommender systems (RecSys '09)*. New York, NY: ACM; 2009, 237–240.
14. Konstan JA, Riedl J. Recommender systems: from algorithms to user experience. *User Model User-adapt Interact* 2012, 22:101–123.
15. Burke R. Hybrid web recommender systems. In: Brusilovski P, Kobsa A, Nejdl W, eds. *The Adaptive Web*. Berlin Heidelberg: Springer-Verlag; 2007, 377–408.
16. Koren Y, Ave P, Park F. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, NV, 2008, 426–434.
17. Hu Y, Volinsky C, Koren Y. Collaborative filtering for implicit feedback datasets. In: *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM'08)*, Pisa, Italy, 2008, 263–272.
18. Golub GH, Reinsch C. Singular value decomposition and least squares solutions. *Numer Math* 1970, 14:403–420.
19. Cremonesi P, Koren Y, Turrin R. Performance of recommender algorithms on top-n recommendation tasks. In: *Proceedings of the fourth ACM conference on Recommender systems (RecSys'10)*, Barcelona, Spain, 2010.
20. Lee J, Lee D, Lee Y-C, Hwang W-S, Kim S-W. Improving the accuracy of top-n recommendation using a preference model. *Inf Sci* 2016, 348:290–304.
21. Shawe-Taylor J, Cristianini N. *Kernel Methods for Pattern Analysis*, vol. 47. New York, NY: Cambridge University Press; 2004.
22. Shani G, Gunawardana A. Evaluating recommendation systems. In: Ricci F, Rokach L, Shapira B, Kantor PB, eds. *Recommender Systems Handbook*. New York: Springer; 2011, 257–298.
23. Hidasi B, Tikk D. General factorization framework for context-aware recommendations. *Data Min Knowl Discov* 2015, 30:342. doi:10.1007/s10618-015-0417-y
24. Said A, Bellogn A. Comparative recommender system evaluation. In: *Proceedings of the 8th ACM Conference on Recommender systems (RecSys'14)*, Foster City, USA, 2014, 129–136.
25. Doerfel S, Group KDE. The role of cores in recommender benchmarking for social bookmarking systems. *ACM Trans Intell Syst Technol* 2016, 7. Available at http://dl.acm.org/citation.cfm?doid=2885506.2700485.
26. Brand M. Incremental singular value decomposition of uncertain data with missing values. In: *Computer Vision—ECCV 2002*. London: Springer-Verlag; 2002, 707–720.
27. Furnas GW, Deerwester S, Dumais ST, Landauer TK, Harshman RA, Streeter LA, Lochbaum KE. Information retrieval using a singular value decomposition model of latent semantic structure. In: *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY: ACM; 1988, 465–480.
28. Zha H, Simon H. On updating problems in latent semantic indexing. *SIAM J Sci Comput* 1999, 21:1–9.
29. Bazire M, Brézillon P. Understanding context before using it. In: Akman V, Bouquet P, Thomason R, Young RA, eds. *Modeling and Using Context*. Berlin Heidelberg: Springer-Verlag; 2005, 29–40.
30. Dourish P. What we talk about when we talk about context. *Pers Ubiquit Comput* 2004, 8:19–30.
31. Adomavicius G, Mobasher B, Ricci F, Tuzhilin A. Context-aware recommender systems. *AI Mag* 2011, 32:67–80.
32. Karatzoglou A, Amatriain X, Baltrunas L, Oliver N. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys'10)*, Barcelona, Spain, 2010, 79.
33. Rettinger A, Wermser H, Huang Y, Tresp V. Context-aware tensor decomposition for relation prediction in social networks. *Soc Netw Anal Min* 2012, 2:373–385.
34. Woerndl W, Schlichter J. Introducing context into recommender systems. In: *Proceedings of the AAAI Workshop on Recommender Systems in e-Commerce*, Vancouver, Canada, 2007.
35. Comon P. Tensors: a brief introduction. *IEEE Signal Process Mag* 2014, 31:44–53.
36. Grasedyck L, Kressner D, Tobler C. A literature survey of low-rank tensor approximation techniques. *GAMM Mitteilungen* 2013, 36:53–78.
37. Kolda TG, Bader BW. Tensor decompositions and applications. *SIAM Rev* 2009, 51:455–500.
38. Eckart C, Young G. The approximation of one matrix by another of lower rank. *Psychometrika* 1936, 1:211–218.
39. De Silva V, Lim L-H. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J Matrix Anal Appl* 2008, 30:1084–1127.

40. Oseledets IV. Tensor-train decomposition. *SIAM J Sci Comput* 2011, 33:2295–2317.

41. Grasedyck L. Hierarchical singular value decomposition of tensors. *SIAM J Matrix Anal Appl* 2010, 31:2029–2054.

42. Chapelle O, Wu M. Gradient descent optimization of smoothed information retrieval metrics. *Inf Retr Boston* 2010, 13:216–235.

43. De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. *SIAM J Matrix Anal Appl* 2000, 21:1253–1278.

44. Rafailidis D, Daras P. The TFC model: tensor factorization and tag clustering for item recommendation in social tagging systems. *IEEE Trans Syst Man Cybern A Syst Hum* 2013, 43:673–688.

45. Sun J-T, Zeng H-J, Liu H, Lu Y, Chen Z. CubeSVD: a novel approach to personalized web search. In: *Proceedings of the 14th International Conference on World Wide Web (WWW'05)*, Chiba, Japan, 2005, 382.

46. Symeonidis P, Nanopoulos A, Manolopoulos Y. A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis. *IEEE Trans Knowl Data Eng* 2010, 22:179–192.

47. Symeonidis P. ClustHOSVD: item recommendation by combining semantically enhanced tag clustering with tensor HOSVD. *IEEE Trans Syst Man Cybern Syst* 2015, 46:1240-1251.

48. Zhang W, Sun H, Liu X, Guo X. An incremental tensor factorization approach for web service recommendation. In: *2014 I.E. International Conference on Data Mining Workshop (ICDMW)*, Shenzhen, China, 2014, 346–351.

49. Hackbusch W. *Tensor Spaces and Numerical Tensor Calculus*, vol. 42. Berlin Heidelberg: Springer-Verlag; 2012.

50. Navasca C, De Lathauwer L, Kindermann S. Swamp reducing technique for tensor decomposition. In: *Signal Processing Conference, 2008 16th European*. Lausanne: IEEE; 2008, 1–5.

51. Kolda T, Bader B. The TOPHITS model for higher-order web link analysis. *Work Link Anal Counterterrorism Secur* 2006, 7:26–29.

52. Rendle S, Marinho LB. Learning optimal ranking with tensor factorization for tag recommendation. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009, 727–736.

53. Shi Y, Karatzoglou A, Baltrunas L, Larson M, Hanjalic A, Oliver N. TFMAP: optimizing MAP for top-n context-aware recommendation. In: *Proceedings of the 35th ACM Special Interest Group on Information Retrieval (SIGIR)*, Portland, Oregon, 2012, 155–164.

54. Rendle S. Schmidt-Thieme L. Pairwise interaction tensor factorization for personalized tag recommendation. In: *Proceedings of the third ACM International Conference on Web Search and Data Mining (WSDM'10)*, New York, NY, 2010, 81–90.

55. Allan J, Croft B, Moffat A, Sanderson M, Aslam J, Azzopardi L, Belkin N, Borlund P, Bruza P, Callan J, et al. Frontiers, challenges, and opportunities for information retrieval: report from SWIRL 2012 the second strategic workshop on information retrieval in lorne. *SIGIR Forum* 2012, 46:2–32.

56. Kolda TG, Bader BW, Kenny JP. Higher-order web link analysis using multilinear algebra. In: *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, Houston, Texas 2005, 242–249.

57. Cichocki A, Zdunek R, Phan AH, Amari SI. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. Wiley, 2009.

58. Zhou G, Cichocki A, Zhao Q, Xie S. Nonnegative matrix and tensor factorizations: an algorithmic perspective. *IEEE Signal Process Mag* 2014, 31:54–65.

59. Chi Y Zhu S. FacetCube. In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10)*, Toronto, Canada, 2010, 569.

60. Chi Y, Zhu S, Gong Y, Zhang Y. Probabilistic polyadic factorization and its application to personalized recommendation. In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*, Napa Valley, California, Lausanne, Switzerland, 2008, 941–950.

61. Marinho LB, Nanopoulos A, Thieme-Schmidt L, Jaschke R, Hotho A, Stumme G, Symeonidis P. Social tagging recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB, eds. *Recommender Systems Handbook*. New York, NY: Springer-Verlag; 2010, 615–644.

62. Symeonidis P, Symeonidis P, Nanopoulos A, Nanopoulos A, Manolopoulos Y, Manolopoulos Y. Tag recommendations based on tensor dimensionality reduction. In: *Proceedings of the 2008 ACM conference on Recommender systems*, Lausanne, Switzerland, 2008, 7.

63. Symeonidis P, Ruxanda M, Nanopoulos A, Manolopoulos Y. Ternary semantic analysis of social tags for personalized music recommendation. In: *Proceedings of 9th International Conference on Music Information Retrieval*, vol 8, Philadelphia, Pennsylvania, 2008, 219–224.

64. Symeonidis P. User recommendations based on tensor dimensionality reduction. *IFIP Int Fed Inf Process* 2009, 296:331–340.

65. Ross DA, Lim J, Lin R-S, Yang M-H. Incremental learning for robust visual tracking. *Int J Comput Vis* 2008, 77:125–141.

66. Rendle S, Freudenthaler C, Gantner Z, Schmidt-thieme L. BPR: Bayesian personalized ranking from implicit feedback. In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI'09)*, Quebec, Canada, 2009, 452–461.
67. Liao Z, Wang C, Zhang M. A tripartite tensor decomposition fold- in for social tagging. *J Appl Sci Eng* 2014, 17:363–370.
68. Zhang M, Ding C, Liao Z. Tensor fold-in algorithms for social tagging prediction. In: *Proceedings of the 2011 I.E. 11th International Conference on Data Mining*, Vancouver, Canada, 2011, 1254–1259.
69. Gantner Z, Rendle S, Schmidt-Thieme L. Factorization models for context-/time-aware movie recommendations. In: *Proceedings of the 2nd International Workshop on Context-Aware Recommender Systems*, Barcelona, Spain, 2010, 14–19.
70. Hidasi B. Factorization models for context-aware recommendations. *Infocommun J* 2014, VI:27–34.
71. Shan L, Lin L, Sun C, Wang X. Predicting ad click-through rates via feature-based fully coupled interaction tensor factorization. *Electron Commer Res Appl* 2016, 16:30–42.
72. Zhao X, Li X, Liao L, Song D, Cheung WK. Crafting a time-aware point-of-interest recommendation via pairwise interaction tensor factorization. *Knowl Sci Eng Manage* 2015, 9403:458–470.
73. Wermser H, Rettinger A, Tresp V. Modeling and learning context-aware recommendation scenarios using tensor decomposition. In: *2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011)*, Kaohsiung, Taiwan, 2011, 137–144.
74. Xu Y, Zhang L, Liu W. Cubic analysis of social bookmarking for personalized recommendation. In: *Proceedings of the 8th Asia-Pacific Web conference on Frontiers of WWW Research and Development*, Harbin, China, 2006, 733–738.
75. Fu X, Huang K, Ma W-K, Sidiropoulos ND, Bro R. Joint tensor factorization and outlying slab suppression with applications. *Signal Process IEEE Trans* 2015, 63:6315–6328.
76. Nanopoulos A, Rafailidis D, Symeonidis P, Manolopoulos Y. MusicBox: personalized music recommendation based on cubic analysis of social tags. *IEEE Trans Audio Speech Lang Process* 2010, 18:407–412.
77. Wang L, Meng X, Zhang Y. Applying HOSVD to alleviate the sparsity problem in context-aware recommender systems. *Chin J Electron* 2013, 22:773–778.
78. Xiong L, Chen X, Huang T-k, Schneider J, Carbonell JG. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In: *Proceedings of the SIAM International Conference on Data Mining*, Columbus, Ohio, 2010, 211–222.
79. Wang Z, He L. User identification for enhancing IP-TV recommendation. *Knowl-Based Syst* 2016, 98:68–75.
80. Hidasi B, Tikk D. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In: *ECML PKDD'12 Proceedings of the 2012 European conference on Machine Learning and Knowledge Discovery in Databases*, Bristol, UK, 2012, 67–82.
81. Hidasi B, Tikk D. Context-aware recommendations from implicit data via scalable tensor factorization. arXiv Prepr. arXiv1309.7611; 2013.
82. Rendle S, Gantner Z, Freudenthaler C, Schmidt-Thieme L. Fast context-aware recommendations with factorization machines. In: *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY: ACM; 2011, 635–644.
83. Kutty S, Chen L, Nayak R. A people-to-people recommendation system using tensor space models. In: *Proceedings of the 27th ACM International Symposium on Applied Computing (SAC'12)*, Trento, Italy, 2012, 187.
84. Sizov S, Staab S, Franz T. *Analysis of Social Networks by Tensor Decomposition*. New York: Springer; 2010, 45–58.
85. Barmpoutis P, Kotropoulos C, Pliakos K. Image tag recommendation based on novel tensor structures and their decompositions. In: *2015 9th International Symposium on Image Signal Processing and Analysis (ISPA)*. Zagreb: IEEE; 2015, 7–12.
86. Panagopoulos M, Kotropoulos C. Image tagging using tensor decomposition. In: *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, Corfu, Greece, 2015.
87. Dunlavy DM, Kolda TG, Kegelmeyer WP. Multilinear algebra for analyzing data with multiple linkages. *Graph Algorithms Lang Linear Algebr* 2011, 2006:85–114.
88. Yao L, Sheng QZ, Qin Y, Wang X, Shemshadi A, He Q. Context-aware point-of-interest recommendation using tensor factorization with social regularization. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Santiago, Chile, 2015, 1007–1010.
89. Ma H, Zhou D, Liu C, Lyu MR, King I. Recommender systems with social regularization. In: *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*. New York, NY: ACM; 2011, 287–296.
90. Misaghian N, Jalali M, Moattar MH. Resource recommender system based on tag and time for social tagging system. In: *Proceedings of the 3rd International Conference on Computer and. Knowledge*

Engineering (ICCKE 2013), Mashhad, Iran, 2013, 97–101.

91. Rafailidis D, Nanopoulos A. Modeling the dynamics of user preferences in coupled tensor factorization. In: *Proceedings of the 8th ACM Conference on Recommender systems (RecSys'14)*. Foster City, California2014, 321–324.
92. Acar E, Kolda TG, Dunlavy DM. All-at-once optimization for coupled matrix and tensor factorizations. arXiv Prepr. arXiv1105.3422; 2011.
93. Ermis B, Acar E, Cemgil AT. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Min Knowl Discov* 2013, 29:203–236.
94. Bhargava P, Phan T, Zhou J, Lee J. Who, what, when, and where: multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In: *Proceedings of the 24th International Conference on World Wide Web*, Florence, Italy, 2015, 130–140.
95. Yang Z, Yin D, Davison BD. Recommendation in academia: a joint multi-relational model. In: *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. Beijing: IEEE; 2014, 566–571.
96. Wang Y, Liu Y, Yu X. Collaborative filtering with aspect-based opinion mining: a tensor factorization approach. In: *Proceedings of the 2012 I.E. 12th International Conference on Data Mining*, Brussels, Belgium, 2012, 1152–1157.
97. Adomavicius G, Manouselis N, Kwon Y. Multi-criteria recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB, eds. *Recommender Systems Handbook*. New York: Springer; 2011, 769–803.
98. Nilashi M, Ibrahim OB, Ithnin N. Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and Neuro-Fuzzy system. *Knowl-Based Syst* 2014, 60:82–101.
99. Symeonidis P, Krinis A, Manolopoulos Y. GeoSocialRec: explaining recommendations in location-based social networks. *Adv Databases Inf Syst* 2013, 8133:84–97.
100. Symeonidis P, Papadimitriou A, Manolopoulos Y, Senkul P, Toroslu I. Geo-social recommendations based on incremental tensor reduction and local path traversal. In: *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks (LBSN'11)*, Chicago, Illinois, 2011, 1.
101. Sattari M, Toroslu IH, Karagoz P, Symeonidis P, Manolopoulos Y. Extended feature combination model for recommendations in location-based mobile services. *Knowl Inf Syst* 2014, 44:629–661.
102. Chen W, Hsu W, Lee ML. Making recommendations from multiple domains. In: *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*. New York: ACM Press; 2013, 892.
103. Hu L, Cao J, Xu G, Cao L, Gu Z, Zhu C. Personalized recommendation via cross-domain triadic factorization. In: *Proceedings of the 22nd International Conference on World Wide Web*, Rio de Janeiro, Brazil, 2013, 595–605.
104. Goreinov SA, Tyrtyshnikov EE, Zamarashkin NL. A theory of pseudoskeleton approximations. *Linear Algebra Appl* 1997, 261:1–21.
105. Oseledets I, Tyrtyshnikov E. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl* 2010, 432:70–88.
106. Oseledets IV, Savostianov DV, Tyrtyshnikov EE. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM J Matrix Anal Appl* 2008, 30:939–956.
107. Mahoney MW, Maggioni M, Drineas P. Tensor-CUR decompositions for tensor-based data. *SIAM J Matrix Anal Appl* 2008, 30:957–987.

