Metalearning for Context-aware Filtering: Selection of Tensor Factorization Algorithms

Tiago Cunha
Faculdade de Engenharia da
Universidade do Porto
Rua Dr. Roberto Frias
Porto, Portugal 4200-465
tiagodscunha@fe.up.pt

Carlos Soares
Faculdade de Engenharia da
Universidade do Porto
Rua Dr. Roberto Frias
Porto, Portugal 4200-465
csoares@fe.up.pt

André C.P.L.F. de Carvalho
Universidade de São Paulo, ICMC
Rua Trabalhador Sancarlense
São Carlos, São Paulo, Brasil
andre@icmc.usp.br

ABSTRACT

This work addresses the problem of selecting Tensor Factorization algorithms for the Context-aware Filtering recommendation task using a metalearning approach. The most important challenge of applying metalearning on new problems is the development of useful measures able to characterize the data, i.e. metafeatures. We propose an extensive and exhaustive set of metafeatures to characterize Context-aware Filtering recommendation task. These metafeatures take advantage of the tensor's hierarchical structure via slice operations. The algorithm selection task is addressed as a Label Ranking problem, which ranks the Tensor Factorization algorithms according to their expected performance, rather than simply selecting the algorithm that is expected to obtain the best performance. A comprehensive experimental work is conducted on both levels, baselevel and metalevel (Tensor Factorization and Label Ranking, respectively). The results show that the proposed metafeatures lead to metamodels that tend to rank Tensor Factorization algorithms accurately and that the selected algorithms present high recommendation performance.

CCS CONCEPTS

•**Information systems → Recommender systems;** *Data mining;* •**Computing methodologies → Machine learning;**

KEYWORDS

Context-aware Filtering, Tensor Factorization, Metalearning, Algorithm Selection, Label Ranking

1 INTRODUCTION

Recommender Systems (RSs) deal with the information overload problem by recommending potentially interesting items to users [3]. Several recommendation strategies exist, including Collaborative Filtering (CF) and Context aware Filtering (CAF). In this work, we focus on CAF recommendations, more specifically regarding the usage of Tensor Factorization (TF) algorithms.

Despite the large amount of research dedicated to RS, there are still several challenges that need to be addressed. One of them is how to choose the best CAF algorithm(s) for a given dataset. This study investigates the use of a Metalearning (MtL) approach for CAF algorithm selection. Although the area of algorithm selection for RS has been growing [5, 9, 14, 16, 22, 29], to the best of our knowledge, CAF algorithm selection has never been addressed.

MtL approaches to the algorithm selection problem consist of describing the datasets (i.e. tensors) with quantitative characteristics (i.e. metafeatures) and the relative performance of a set of algorithms. This creates the so called metadata. Learning algorithms can then be applied to this metadata to extract patterns and used them for the selection of the best algorithm(s) for new datasets/problems. MtL approaches are thus organized in base- and metalevels. In this work, the baselevel refers to the CAF recommendation problem using TF and the metalevel is related to the selection of the best ranking of algorithm using Label Ranking (LR).

One of the main challenges of developing MtL approaches is the design of the metafeatures. Metafeatures must 1) contain information about the data that affects the behavior of algorithms and 2) be computationally efficient [7]. Although MtL approaches for CF algorithms have been published [5, 9, 14, 16, 22, 29], given the difference in the structure of the data (matrices vs tensors), a completely new set of metafeatures is required for CAF. Another important issue addressed in this work is related to the type of MtL task. Often, MtL problems are addressed as classification tasks. This means that, given a new problem, a single algorithm is selected. However, it is well-known that this approach has important shortcomings, namely, when the selected algorithm fails, the user is left unsupported [7]. Hence, MtL for algorithm selection should be addressed as a LR task, in which the output is a ranking of candidate algorithms [7]. Thus, the most important contributions are:

- **Label ranking approach to MtL:** metamodels are created using LR to predict a ranking of TF algorithms.
- **Metafeatures for tensors designed using a systematic framework:** tensors are described in an extensive and exhaustive fashion by taking advantage of the tensor's hierarchical structure and slice operations.
- **Interpretation of the metamodel:** the importance of the metafeatures in the performance of the metamodel for selection of TF algorithms is analyzed.

This document is organized as follows: Section 2 presents the theoretical background on all areas involved in this work; Section 3 devotes itself to the proposal of metafeatures for tensors, while Section 4 presents the experimental work undertaken at both the

RecSys'17, August 27–31, 2017, Como, Italy
© 2017 ACM. ISBN 978-1-4503-4652-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3109859.3109899

base- and metalevels. The results and their analysis are presented in Section 5, while the conclusions are summarized in Section 6.

2 RELATED WORK

2.1 Recommender Systems

RSs attempt to extract patterns from data, which have the potential to explain and predict recommendations of new items. There are, however, several different strategies to make said recommendations. This work focuses on CF and its extension for CAF through TF.

CF recommendations are based on the premise that a user will probably like the items favored by a similar user. CF employs the feedback from each individual user to recommend items to similar users [50]. Such feedback is represented in a rating matrix, a data structure which is described as $R^{U \times I}$, representing a set of users U, where $u \in \{1, ..., N\}$ and a set of items I, where $i \in \{1, ..., M\}$. Each element of this matrix (R_{ui}) is the feedback provided by user u for item i. The recommendations are typically achieved via Nearest Neighbor [12, 37] or Matrix Factorization techniques [25, 31].

CAF is an extension of CF that uses additional information, for instance time or location, to increase the accuracy of the recommendations [2, 6]. The motivation lies in the premise that the contextual dimensions contains additional useful information to the information in the user and item dimensions in CF. For instance, taking into account the time of the day, the recommendation may capture the difference in music styles a user listens to at work and at home. There are several ways to incorporate contextual information in RS [4], but we focus on the combination of CF and CAF.

Using context in recommendations requires a change in paradigm. The relationship $R : User \times Item \rightarrow Rating$, is now represented as $R : User \times Item \times Context \rightarrow Rating$, where $User$ and $Item$ refer to the user and item domains and $Context$ is related to the contextual domain [4]. This change in paradigm is essential to understand why tensors are the ideal data structure for CAF: in the same way the CF paradigm required a matrix (i.e. bidimensional array or 2-order tensor) to hold the feedback values, the most natural representation for the newer paradigm is a 3-order tensor (i.e. multidimensional array).

2.2 Tensor Factorization

The first step for TF is tensor decomposition. This refers to the the way the original tensor is modeled in an intermediary step before the application of the factorization algorithm. The two most popular decomposition methods are CANDECOMP/PARAFAC (CP) [8, 18] and Tucker [45]. Despite the existence of other methods [24], we focus only on these since these are the only ones used in CAF.

The CP decomposition factorizes a tensor into a sum of component rank-one tensors (see Figure 1). Being a rank-one tensor means it can be written as the outer product of N vectors, i.e. $\mathcal{X} = a(1) \circ a(2) \circ ... \circ a(N)$, where $\circ$ represents the outer product. The CP decomposition of a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is [24]:

$$\mathcal{X} \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r \tag{1}$$

where R is a positive integer and $a_r \in \mathbb{R}^I$, $b_r \in \mathbb{R}^J$ and $c_r \in \mathbb{R}^K$.

The Tucker decomposition transforms a tensor into a core tensor multiplied by a matrix along each mode [24]. Therefore, the decomposition of a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ (see Figure 2) is:

$$\mathcal{X} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_p \circ b_q \circ c_r \tag{2}$$

where $A \in \mathbb{R}^{I \times P}$, $B \in \mathbb{R}^{J \times Q}$ and $C \in \mathbb{R}^{K \times R}$ are the factor matrices and $G \in \mathbb{R}^{P \times Q \times R}$ is the core tensor. The operator $\times_N$ refers to the N-mode product between the tensor and a matrix.

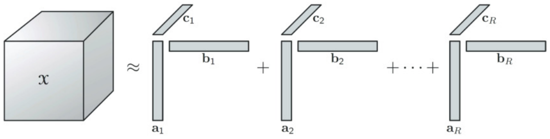


Figure 1: CP decomposition [24].

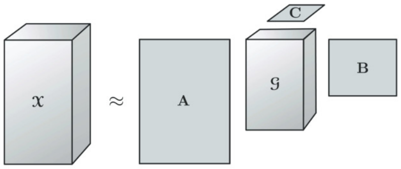


Figure 2: Tucker decomposition [24].

Different decompositions require different algorithms to perform the factorization. While CP is typically approached using Alternating Least Squares (ALS), Tucker decomposition takes advantage of Higher Order Singular Value Decomposition (HOSVD) [24].

2.3 Context aware Filtering with Tensors

CAF using TF has been widely used in the context of RS. The first approaches to apply TF to RS used a HOSVD algorithm and Tucker decomposition, but with different contextual dimensions: time [23] and semantic tags [42, 43]. Recent papers shifted towards the usage of implicit feedback and to replace Tucker by CP decompositions. While some focused the optimization of Mean Average Precision as the loss function [39], others have adapted ALS to perform the factorization [21] and even proposed Bayesian Probabilistic approaches to model the factorization [49]. Some works look at the problem in non-conventional ways: either by applying Factorization Machines instead of TF algorithms [34] or by changing the tensor to a so called Contextual Operating Tensor [28].

2.4 Metalearning

Metalearning (MtL) is concerned with relating characteristics of the data with the behavior of algorithms [46]. It has been extensively used for algorithm selection [27, 33, 35, 36, 41]. The algorithm selection task can be viewed as a learning problem itself, by casting it as a predictive task. For such, it uses a metadataset, where each meta-example corresponds to a dataset. For each meta-example, the predictive features are characteristics (metafeatures) extracted from the corresponding dataset and the targets are the performance (metalabels) of algorithms when applied to the dataset [7].

One of the main challenges in MtL is to define which are the metafeatures that effectively describe how strongly a dataset matches the bias of each algorithm [7]. The MtL literature divides the metafeatures into three main groups [7, 38, 46]:

- statistical and/or information-theoretical measures (obtain numerical data descriptors using standard formulas);
- landmarking-based properties (performance measures obtained from the fast application of algorithms to the datasets);
- model-based characteristics (extraction of properties from fast/simplified models induced from the datasets).

As far as we know, there is no prior work on the algorithm selection for tensors in CAF. There are, however, a few approaches to select CF algorithms [5, 9, 14, 16, 22, 29], which we use as guidelines. Specifically, we inspire ourselves on the statistical and/or information-theoretical measures used in the related work to describe the tensor. However, we must point out the significant gap in difficulty between these two tasks: while in CF one only looks towards describing a matrix, in CAF a much more complex structure is tackled: the tensor. This requires a completely novel and more complex approach. To organize the metafeatures in a more interpretable way, we use a framework [32], described next.

2.5 Systematic metafeatures framework

The systematic metafeatures framework [32] requires three main elements: objects o, functions f and post-functions pf. The process applies each function to each object and, afterwards, each post-function to that outcome in order to generate a single metafeature value. Thus, this framework is represented as: $\{o\}.\{f\}.\{pf\}$. An example is the average sum of ratings for all users.

One important property of this framework is its recursiveness. If one poses the problem in this way, then the outcome of an inner level (IL) application of the framework can be used as the result of an outer level (OL) function. This means that the generic description $\{o\}.\{f\}.\{pf\}$ can be transformed into:

$$\begin{aligned}\{\text{OL-}o\}.\{\text{OL-}f\}.\{\text{OL-}pf\} = \\ \{\text{OL-}o\}.\Big[\{\text{IL-}o\}.\{\text{IL-}f\}.\{\text{IL-}pf\}\Big].\{\text{OL-}pf\}\end{aligned} \tag{3}$$

3 METAFEATURES FOR TENSORS

3.1 Tensor basics

Consider a third-order tensor $T \in \mathbb{R}^{U \times I \times C}$ with the dimensions U, I and C, referring specifically to the User, Item and Context dimensions [24]. The dimensions can be characterized as a sequence of ordered and independent elements: $U = (u_x)_{x=1}^{L}, I = (i_y)_{y=1}^{M}$ and $C = (c_z)_{z=1}^{N}$, with L, M and N are the maximum number of users, items and contexts, respectively.

Note that by considering only the dimensions U and I, one refers to the CF data structure, i.e. the rating matrix. The contextual dimension C is appended to the tensor, hence enabling another equally important decision factor for the recommendation problem. Each dimension is related to one another in an orthogonal basis, meaning that any point in this data structure is simultaneously mapped on all dimensions. This enables multi-dimensional relationships, which motivate the usage of TF algorithms in CAF [2].

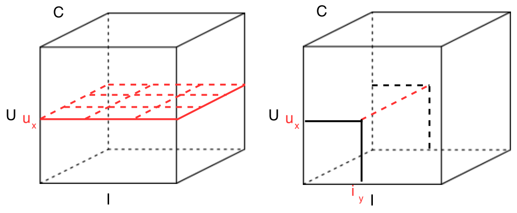
3.2 Tensor slices

In a tensor, data can be queried in several perspectives and with different levels of detail. By selecting specific values on one or more dimensions, one obtains different data structures which contain different patterns. The operation that allows the retrieval of such data structures is known as slice operation [24]. In the context of this work, there are two well defined slice operations, which we name vector-level and matrix-level. The results of such operations are vectors or matrices of ratings, respectively. The slice operations are formalized in Equations 4 and 5 for the matrix and vector-levels, respectively. When an element is not provided, it is represented as $\varnothing$. Considering user u, item i and context c, the different slice operations can be formalized as:

$$\begin{aligned} SLICE(T, u, \varnothing, \varnothing) &= \bigcup_{y=1}^{M} \bigcup_{z=1}^{N} T_{[u, i_y, c_z]} \in \mathbb{R}^{I \times C} \\ SLICE(T, \varnothing, i, \varnothing) &= \bigcup_{x=1}^{L} \bigcup_{z=1}^{N} T_{[u_x, i, c_z]} \in \mathbb{R}^{U \times C} \\ SLICE(T, \varnothing, \varnothing, c) &= \bigcup_{x=1}^{L} \bigcup_{y=1}^{M} T_{[u_x, i_y, c]} \in \mathbb{R}^{U \times I} \end{aligned} \tag{4}$$

$$\begin{aligned} SLICE(T, u, i, \varnothing) &= \bigcup_{z=1}^{N} T_{[u, i, c_z]} \in \mathbb{R}^{C} \\ SLICE(T, u, \varnothing, c) &= \bigcup_{y=1}^{M} T_{[u, i_y, c]} \in \mathbb{R}^{I} \\ SLICE(T, \varnothing, i, c) &= \bigcup_{x=1}^{L} T_{[u_x, i, c]} \in \mathbb{R}^{U} \end{aligned} \tag{5}$$

A representation of the slice operations is shown in Figure 3. Notice that while in Figure 3a, only a specific user u_x is provided, in Figure 3b, an user u_x and an item i_y are both specified. In the first case, the result is a matrix, while the second is a vector.



(a) Matrix-level. (b) Vector-level.

Figure 3: Slice operations.

3.3 Metafeatures overview

We now provide an overview of the application of the systematic metafeatures framework (Section 2.5) in terms of objects, functions and post-functions for CAF. The specific details about the metafeatures proposed are given in the following sections.

3.3.1 Objects. Recall that the framework applies functions to a set of objects. Therefore, it is essential to identify suitable objects. An obvious choice of object is the tensor itself. However, since there are few known descriptors for the data structure as whole, one must look beyond. We assume that the vectors and matrices obtained with slice operations (section 3.2) have potential to affect the baselevel algorithm performance. If such assumption is true, then descriptors of these data structures have potential to positively impact the algorithm selection problem. Here, these metafeatures are complemented with global tensor characteristics, such as the Frobenius norm, in order to understand their importance.

The tensor is hierarchically organized in terms of slice operations, as it is described in Figure 4. Three different granularity levels are presented: tensor, matrix and vector. To access the matrix-level data, one fixes one of the dimensions according to a specific element. However, one can also go a granularity level deeper, by providing one additional element of another dimension, reaching the vector-level. It is also possible to go even further, reaching the specific rating. However, this level can be discarded since it cannot be considered a suitable metafeature. By definition, a metafeature is an unique descriptor, representing an aggregated characteristic of a dataset [7]. Since MtL requires several datasets, each with different amounts of users, items and contexts, the metafeatures extracted must be aggregated to create a unique feature which can be extracted in all tensors. Thus, it does not make sense to compute metafeatures based on single values.

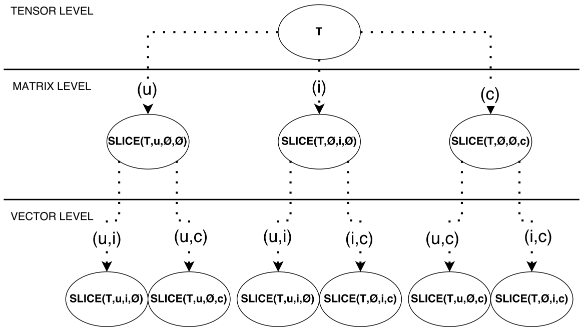


Figure 4: Tensor's structure in terms of slice operations.

Another important aspect of this work is the investigation of relationships among objects and their impact for the selection of algorithms. Hence, besides describing the tensor in several facets, we also attempt to assess the relationships among objects belonging to the same level of slice operations. This approach makes our proposal of metafeatures for CAF a much more difficult problem than it is found in CF [5, 9, 14, 16, 22, 29].

3.3.2 Functions. Following the systematic metafeatures framework, we know that the functions depend on the objects used. Considering the objects studied can be vectors, matrices and the tensor itself, then the task becomes non-trivial. Two properties were identified: the adaptability to the objects and the differentiation between functions that describe a single object or comparisons of objects. The requirements for the different levels in terms of these properties are discussed in Sections 3.4, 3.5 and 3.6.

An important issue concerning our proposal of metafeatures for tensors is that all vectors/matrices to be characterized contain ratings, independently of the type of slice operation employed. This means that since they all share the same data characteristics, one can take advantage of the same functions. This is not true in other MtL scenarios, such as classification and regression [7, 26].

3.3.3 Post-functions. The post-functions to be used must aggregate the result of the application of a function to a set of objects. This means that ideally one should use a large set of statistical properties to describe such outcome. However, due to the combinatorial nature of the framework, this contributes to an explosion of metafeatures to be extracted. Hence, we typically use 3 measures, i.e. one measure for each of the essential elements of univariate analysis: central tendency, dispersion and shape.

3.3.4 Outline. The different metafeatures, organized by level, are described next. The vector-level looks towards describing all the different vectors in a tensor through univariate analysis, yielding structural, statistical and information-theoretical properties. These metafeatures test the assumption that if tensors have different patterns regarding the vectors of one or more dimensions, then the best algorithms will also be different. The matrix-level focuses on describing all the matrices in the tensor through structural and algebraic properties and applies multivariate analysis on the vectors which compose its dimensions. The motivation for these metafeatures lies in testing the assumption that the matrices structure and the relationships among its vectors are informative for the selection of algorithms. This allows to assess whether descriptors of objects or descriptors of interactions among objects hold higher informative power. Lastly, the tensor level extracts structural and algebraic tensor properties and applies multivariate analysis to assess the relationships among its matrices. The motivation for the first is clear: here, we test whether generic tensor descriptors hold informative value which overcomes those of the metafeatures extracted via the remaining slice operations. The second follows the approach also used in the matrix-level.

3.4 Vector-level metafeatures

The vector-level is reached by specific slice operations. We refer to $SLICE(T, u, i, \varnothing)$, $SLICE(T, u, \varnothing, c)$ and $SLICE(T, \varnothing, i, c)$ as **UI**, **UC** and **IC vectors**, respectively. These are the different classes of objects for this metafeatures level.

The functions considered at this level can be categorized according to the properties they represent: structural, statistical and information-theoretical. Structural properties represent the overall dimensions of the vector. We use **size** and **vector sparsity**. Statistical properties characterize the distribution of the ratings. Since this level is concerned with vectors, the metafeatures use functions from univariate analysis [44]. Here, several well defined functions exist: central tendency (**mean**, **mode** and **median**), dispersion (**variance**, **maximum**, **minimum**, **first quartile**, **third quartile** and **standard deviation**) and shape (**skewness** and **kurtosis**). Lastly, one information-theoretical property is used to assess the information within a specific vector: **entropy**.

The post-functions used are: **mean**, **standard deviation** and **skewness**. This creates 126 metafeatures at the vector-level, which can be summarized as:

$$\begin{aligned}&\{\text{UI, UC and IC vectors}\}.\{\text{structural, statistical and}\\&\text{information-theoretical properties}\}.\{\text{post-functions}\}\end{aligned} \tag{6}$$

3.5 Matrix-level metafeatures

The matrix-level slice operations $SLICE(T, u, \varnothing, \varnothing)$, $SLICE(T, \varnothing, i, \varnothing)$ and $SLICE(T, \varnothing, \varnothing, c)$, are referred to here as **IC**, **UC** and **UI matrices**, respectively. Recall the metafeatures in this level are organized into matrix descriptors and descriptors of the pairwise comparison of the matrix vectors. These are explained next.

3.5.1 Matrix metafeatures. To describe a matrix, structural and algebraic properties can be extracted. Such structural properties are **number of rows**, **number of columns**, **number of ratings**, **matrix sparsity** and the **ratio of rows over columns**. In terms of algebraic properties, only the **norm** can be used for non-square matrices. This limits severely the amount and diversity of properties to extract. To obtain additional properties it is possible to transform the matrix into a squared one, using, for instance, similarity or correlation matrices. Then, properties such as trace, determinant and eigenvalues can be used [19].

Other techniques can also be employed at this stage, such as Matrix Factorization and clustering. These can afterwards be described through a wide range of properties, such as the properties of the factorized matrices or the number and size of clusters found. However, a high computational is expected to compute these metafeatures, which makes them unsuitable for algorithm selection purposes.

The post-functions used at this level are: **mean**, **standard deviation** and **skewness**. Note that 3 properties (number of rows, columns and ratio of rows over columns) are the same for all slices of the same dimension. Therefore, these are not aggregated. The 36 metafeatures produced follow this notation:

$$\begin{aligned}&\{\text{UI, UC and IC matrices}\}.\{\text{structural and}\\&\text{algebraic properties}\}.\{\text{post-functions or } \varnothing\}\end{aligned} \tag{7}$$

3.5.2 Vector pairwise comparison. These metafeatures describe the relationships among matrix dimensions based on the vectors it contains. The same vectors were already analyzed independently at the vector-level (Section 3.4). Such relationships are assessed using bivariate analysis techniques which do not require dependent variables, since the data used in CAF does not have this property [44].

These pairwise comparisons can be classified as intra or inter-dimensional: while the first focus on comparing, for instance two vectors of users, the other compares a user vector with a context or item vector. This differentiation is required since the nature of the information each of these vectors represents is different (e.g. the ratings of a single user concerning many items vs the ratings of many users concerning a single item).

Intra-dimensional pairwise comparisons always deal with vectors of the same size. This allows more options in terms of functions to be used. We focus on statistical (**correlation**) and information-theoretical (**mutual information**) properties for these problems. On the other hand, since inter-dimensional pairwise comparisons must deal with vectors of different sizes, the functions used are distances between distributions. In this case, one information-theoretical property is obtained via the **Jensen–Shannon divergence**. The post-functions are: **mean**, **standard deviation** and **skewness**. This yields 15 metafeatures, represented as follows:

$$\begin{aligned}&\{\text{rows, columns}\}.\{\text{correlation, mutual inform.}\}.\{\text{post-functions}\}\\&\{\text{rows/columns}\}.\{\text{Jensen–Shannon divergence}\}.\{\text{post-functions}\}\end{aligned} \tag{8}$$

These metafeatures are computed for all matrices, yielding a very large number of metafeatures, which depends on the size of the dimensions. Since the size of dimensions varies across problem instances, this means that it is not possible to characterize all of them with a vector of metafeatures of the same size. For instance, if we consider all the UI matrices of two tensors, we cannot create assume the amount of features extracted will be the same, since the dimensions can be different. Thus, the framework is applied once more, where the objects are **IC**, **UC** and **UI matrices** and the functions are the outcome of the comparison between vectors. The post-functions used are: **mean**, **standard deviation** and **skewness**. The resulting 135 metafeatures follow this notation:

$$\{\text{UI, UC and IC matrices}\}.\{\text{pairwise comparisons}\}.\{\text{post-functions}\} \tag{9}$$

3.6 Tensor-level metafeatures

Lastly, we describe the tensor-level metafeatures. These are split into those that describe the tensor as a whole and those that are based on pairwise comparisons between its constituent matrices. While the former do not require slice operations, the latter use them to obtain the matrices to be used in the calculations.

3.6.1 Tensor metafeatures. The tensor, which is the object to be processed in the metafeatures framework at this level, can be described by several structural properties: **number of users**, **number of items**, **number of contexts**, **number of ratings**, **tensor sparsity**, **proportion of users over items**, **proportions of users over contexts** and **proportions of items over contexts**. An algebraic operation, the **Frobenius norm**, is also used.

Similarly to Section 3.5.1, other techniques can also be used at this level. The usage of, for instance, multilinear PCA and clustering may extract other types of metafeatures representing different types of patterns that affect the behavior of algorithms. However, the computational cost of these methods make them unsuitable for the algorithm selection problem. Unlike all other metafeatures, these output a single value and, thus, do not require the use of post-functions. Therefore, a total of 9 metafeatures can be formulated:

$$\{\text{tensor}\}.\{\text{structural and algebraic properties}\}.\varnothing \tag{10}$$

3.6.2 Matrix pairwise comparison. The pairwise comparison of matrices uses slice operations to generate three types of objects: **UI**, **UC** and **IC matrices**. Although we can distinguish the intra and inter-dimensional comparisons of matrices (i.e. comparing matrices of the same or different nature), as before, we use the same function: **correlation**. We obtain the correlation between matrices by calculating the correlation between rows of one matrix and columns of another. The result is a matrix of correlation coefficients. Therefore, the only requirements is that the number of rows of the first matrix matches the number of columns of the second one. Due to the nature of the tensor, all matrices have at least one dimension in common. Hence, it is always possible to apply the correlations to any pair of matrices retrieved from the tensor, even if it may be necessary to transpose one or both matrices. Unlike the previous cases, where the same post-functions have been repeated, here we extract only the **maximum** value for each pairwise comparison. The rationale lies in the assumption that only high dependency matters, unlike the previous examples where an overall perspective is essential. The formalization of all 6 metafeatures is:

$$\begin{gathered}\{\text{UI, UC and IC matrices}\}.\{\text{correlation}\}.\{\text{maximum}\}\\ \{\text{UI/UC, UI/IC and UC/IC matrices}\}.\{\text{correlation}\}.\{\text{maximum}\}\end{gathered} \tag{11}$$

Also as in the case of matrix-level metafeatures, the final set of metafeatures is obtained by applying the framework to the intermediate results. In this case, the object is the **tensor** and the function consists of the results of the comparison between matrices. In this case, we obtain a vector of maximum correlations for intra and inter-dimensional comparisons. The results are aggregated using **mean**, **standard deviation** and **skewness**. The formalization of these 18 metafeatures is:

$$\{\text{tensor}\}.\{\text{pairwise comparisons}\}.\{\text{post-functions}\} \tag{12}$$

4 ALGORITHM SELECTION FOR TF

4.1 Baselevel experimental setup

The baselevel is characterized by the datasets, algorithms, evaluation protocol and evaluation measures. The datasets need to be built in such a way that they can be encapsulated in a tensor. Since we are unaware of public datasets specifically for this purpose, we chose to adapt CF datasets. Hence, we searched the available datasets for a common dimension, which could be used as context. We found that the temporal dimension fitted this criteria, since most datasets have the timestamp of the feedback event.

Dimensions must be nominal variables and, since time is a continuous variable, we discretized it. The temporal dimension is divided into a predefined number of bins and each rating event is assigned to the respective bin. The contextual dimension can therefore be described as the sequence of bins. The number of bins used for dimension C are $\{2, 3, 4, 5, 10, 20, 30, 50, 100\}$. Other measurement units could be used, such as day or hour, but we decided to use an approach which could be easily replicated across datasets and that enabled us to have fixed length for the contextual dimension. A total of 34 CF datasets are used from the following domains: Amazon Reviews [30], ConcertTweets [1], MovieLens [17], MovieTweetings [13], Tripadvisor [48] and Yelp [51]. With different binning parameters, this yields 306 datasets.

In terms of TF, this work dwells with CP decomposition (CPD) with several factorization algorithms: CPD, CPD_ALS, CPD_NLS, CPD_RBS and CPD_MINF. The difference between the algorithms lies in the way the optimization process is performed. The CPD_ALS, CPD_MINF, CPD_NLS and CPD_RBS algorithms are optimization-based and use alternating least squares, nonlinear unconstrained optimization, nonlinear least squares and randomized block sampling, respectively. The default version, CPD, is a high-level algorithm which automatically computes an initialization and uses this as a starting point for one or more optimization steps. The implementations from TensorLab [47] was used. In addition, we implemented a baseline algorithm which is an adaptation of the MostPopular version from the CF framework MyMediaLite [15] applied to tensors. The algorithm works by sorting the item ratings for a given user and context to provide the recommendations. This is considered a baseline, since no factorization is involved.

The chosen evaluation protocol is 10-fold cross-validation. The folds are created by splitting the entire dataset of 4-tuples (user, item, context, rating) into ten different groups. Afterwards, each of the nine possible combinations of groups are used to build the tensor and the remaining one to use in the evaluation stage. Since the recommendation follows a Top-N approach [20], the evaluation measures used are Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Area Under the Curve (AUC). While the first three are devoted to ranking accuracy, AUC assesses classification accuracy.

4.2 Metalevel experimental setup

The results obtained at the baselevel, as described in the previous section, are used to create a ranking of the algorithms for each dataset. This ranking is the metatarget (i.e. the dependent variable at the metalevel). Given that baselevel algorithms are evaluated according to multiple measures, a metatarget is generated for each measure. Thus, we have, in fact, multiple metalearning problems, one for each metatarget. The metafeatures for each dataset are also computed, according to the description in Section 3. The values of the metafeatures are the independent variables of the metadata.

Since the goal is to predict a ranking of algorithms, the metalearning problem is addressed as a LR task. The algorithms used at the metalevel (i.e. meta-algorithms) are: kNN and Naive Bayes (NB) [40], Ranking Tree (RT) [10] and the baseline average ranking (AVG). The accuracy of the predicted rankings is measured by computing their correlation to the target ranking, as is usual in LR. The correlation measure used here is Kendall's Tau coefficient. The metalevel performance is estimated using 10-fold cross-validation.

5 RESULTS AND DISCUSSION

5.1 Metalevel evaluation

The results for the performance of the meta-algorithms in all metatargets in terms of Kendall's Tau coefficient are presented in Figure 5. The results show:

- kNN and NB are always the best and worst meta-algorithms. RT beats the baseline only in NDCG and MAP.
- AUC is the metatarget with the worst overall performance, yielding the worst overall correlations coefficients. The performance in NDCG, MAP and MRR metatargets is fairly similar and high, except for the RT meta-algorithm.
- Since kNN is always above the baseline, one can state that our MtL approach is able to predict the rankings of

algorithms. The results are specially important considering the high correlations for the baseline in 3 metatargets.

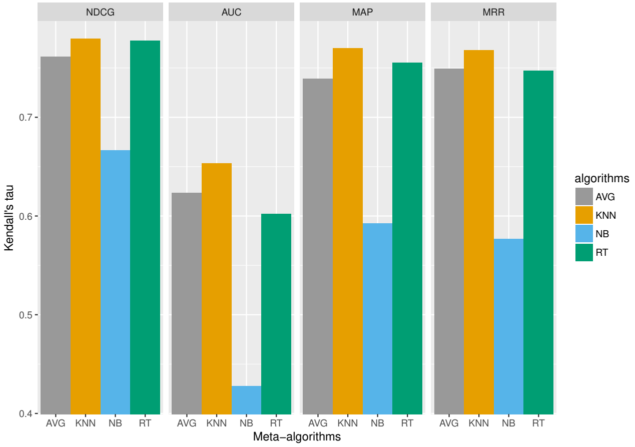


Figure 5: Metalevel evaluation for all meta-algorithms.

To assess whether these results are statistically significant, we use Critical Difference (CD) diagrams [11]. These diagrams rank several methods based on their performance on several datasets. Here, it is applied to the folds of the 10-fold cross validation procedure used on all 4 metatargets. The diagram ranks the algorithms and connects those with differences that are not significantly different (i.e. less than the critical difference, CD). The results are shown in Figure 6 and validate the observations made regarding the meta-algorithms: kNN is better than the baseline and NB; NB is the worst algorithm by far and RT's unstable behavior makes it not statistical significantly different than both kNN and the baseline.

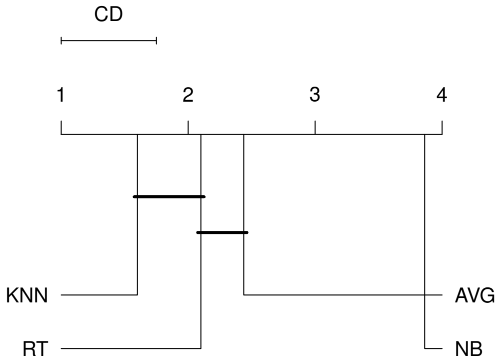
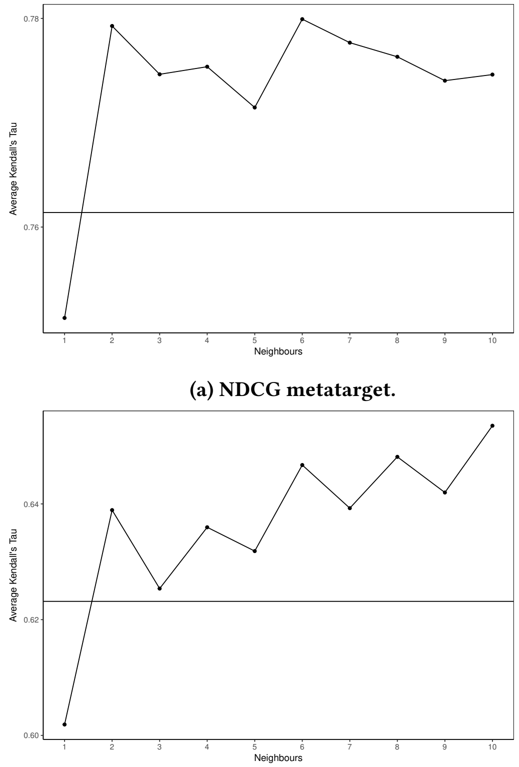


Figure 6: CD diagram for ranking of metamodels.

Based on these results, we further analyzed the behavior of the kNN algorithm. Figure 7 shows Kendall's tau coefficient for $k \in \{1, .., 10\}$ neighbors, separately for the NDCG and AUC metatargets. The baseline is presented in each figure as a horizontal line. Since the results for MRR and MAP are similar to those for NDCG, we do not include them here.

One observes that metamodels beat the baseline for $k > 1$. This shows the robustness of this algorithm for our problem of algorithm selection. However, while for NDCG the performance is fairly constant, in AUC we observe that the performance improves with k. Such variations are dictated by the ranking of algorithms used for training the metamodels, since the metafeatures are the same. This means that the ranking variations in AUC are greater than in NDCG, which makes it a more difficult problem.



(a) NDCG metatarget.

(b) AUC metatarget.

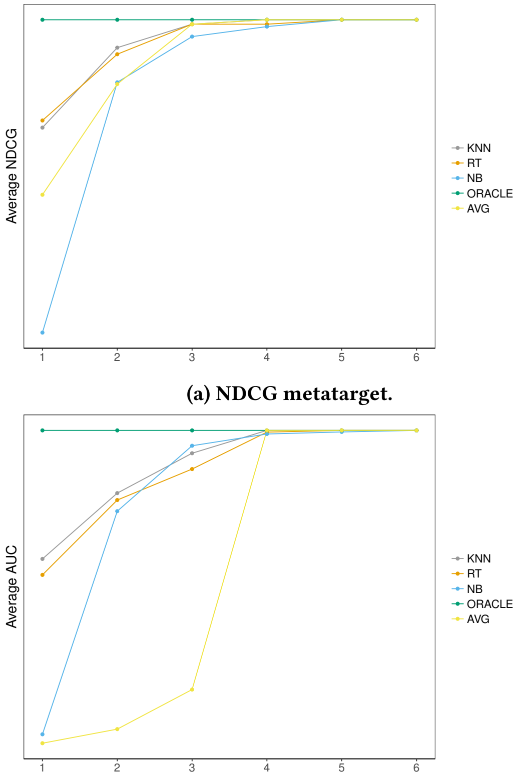
Figure 7: Metalevel evaluation for kNN.

5.2 Baselevel impact

The previous section shows that we can predict the ranking of the algorithms with good accuracy. The question now is whether the use of the algorithms recommended by the metamodel will achieve good baselevel performance. The idea of this evaluation is to simulate different thresholds regarding the number of TF algorithms executed following the ranking order, and to assess the corresponding baselevel performance [7]. A metamodel outperforms the baseline if the performance in a specific threshold is better than the one obtained by average rankings. This mechanism also includes an oracle, which knows beforehand which will be the absolute best algorithm for any case. Therefore, this will create the maximum bound that cannot be beaten (horizontal lines).

Figure 8 presents the baselevel impact for the NDCG and AUC metatargets. Once again, the results for MRR and MAP are not presented as they show the same behavior as NDCG. In both cases is it possible to see that the algorithms recommended by kNN and RT are better than those provided by the baseline recommendation for the top-two algorithms in the predicted rankings. This behavior shows that despite RT not performing well in terms of metalevel evaluation for the AUC and MRR metatargets, the recommendations still yield gains in terms of baselevel performance, on average.

Another interesting observation concerns NB: while for NDCG the performance is worse than the baseline, for AUC it is better. Recalling that this metamodel performed badly in terms of metalevel evaluation, it is surprising to see this behaviour. Once again, this points to the higher difficulty of solving the algorithm selection problem using LR for the AUC metatarget.



(a) NDCG metatarget.

(b) AUC metatarget.

Figure 8: Baselevel impact.

5.3 Importance of metafeatures

Since there is no standard procedure for assessing variable importance in LR, this work uses a simple heuristic approach to address this issue. Using all the metatargets, RT metamodels were trained. Next, the frequency of metafeatures used on those models were computed and ranked. Due to space restrictions, we only summarize the results:

- Most metafeatures are vector pairwise comparisons in nature. Furthermore, their position in the ranking is well established at the top, i.e. 22 out of 24 top metafeatures. This confirms our hypothesis that the characteristics of components of the matrices affect the performance of the CAF methods.
- All other types of metafeatures hold low importance for the problem of algorithm selection. The vector, matrix pairwise comparisons and matrix metafeatures correspond respectively to 6, 2 and 1 of the best metafeatures. This shows the need for future work at these levels.
- More importantly, the metafeatures used to describe the tensor as a whole are not available in the top metafeatures. This, alongside the positive results in both metalevel performance and baselevel impact, validates our approach to use slice operations to create metafeatures for tensors.
- The most represented objects are the UI, IC and UC matrices, respectively with 14, 12 and 7 metafeatures. This is obviously affected by the fact that they are the most frequent type of metafeatures, i.e. the vector pairwise comparison. One notices that IC matrices are represented in 7 out of the top 8 metafeatures.
- The most influential functions are mutual information, correlation and Shannon divergence. There is no clear patterns that distinguishes them, although mutual information is present in 4 of the top 5 metafeatures.

6 CONCLUSIONS AND FUTURE WORK

This work presents the first known approach for algorithm selection for Tensor Factorization algorithms. We follow a Metalearning approach which, given the difference in the nature of the data used in these algorithms in comparison to other algorithm selection tasks previously addressed with metalearning, implies the development of completely new metafeatures to characterize problems. We, thus, propose an extensive and exhaustive set of metafeatures. The metafeatures use slice operations to describe the tensors at several levels of granularity and perspectives. The merits of the proposed approach were validated in a large scale experimental setup. The results have shown that kNN models are capable of solving the problem with statistical significantly better performance than the baseline for all metatargets. In terms of baselevel impact, the models also perform generally better than the baseline. Furthermore, vector pairwise comparison metafeatures have established themselves as the most influential, confirming our hypothesis that these components of the data contain useful information to understand the behavior of TF algorithms. Future work can take several directions: to study contextual dimensions beyond time, include specially designed Tensor Factorization algorithms for Context aware Filtering, study the hyperparameter selection problem, improve and analyze the metafeatures proposed by using more advanced techniques and extend the amount and nature of evaluation measures used in the base- and metalevels.

ACKNOWLEDGEMENTS

This work is financed by the ERDF Fund through the Operational Program for Competitiveness and Internationalization - COMPETE 2020 of Portugal 2020 through the National Innovation Agency (ANI) as part of project 3506. The authors would also like to acknowledge the support from CNPq and FAPESP, Brazilian funding agencies. Special thanks to Hadi Fanaee.

REFERENCES

[1] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. Estimating the Value of Multi-Dimensional Data Sets in Context-based Recommender Systems. In *8th ACM Conference on Recommender Systems (RecSys 2014)*.

[2] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. 2005. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *ACM Transactions on Information Systems* 23, 1 (2005), 103–145.

[3] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowl. and Data Eng.* 17, 6 (2005), 734–749.

[4] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-Aware Recommender Systems. In *Recommender Systems Handbook*, Paul B. Ricci, Francesco and Rokach, Lior and Shapira, Bracha and Kantor (Ed.). Springer US, Chapter 7, 217–253.

[5] Gediminas Adomavicius and Jingjing Zhang. 2012. Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems* 3, 1 (2012), 1–17.

[6] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (2013), 109–132.

[7] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. 2009. *Metalearning: Applications to Data Mining* (1 ed.). Springer Publishing Company, Incorporated.

[8] J Douglas Carroll and Jih-Jie Chang. 1970. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika* 35, 3 (1970), 283–319. DOI: http://dx.doi.org/10.1007/BF02310791

[9] Tiago Cunha, Carlos Soares, and André C.P.L.F. de Carvalho. 2016. Selecting Collaborative Filtering algorithms using Metalearning. In *European Conference on Machine Learning and Knowledge Discovery in Databases.* 393–409.

[10] Cláudio Rebelo de Sá, Carlos Soares, Arno Knobbe, and Paulo Cortez. 2016. Label Ranking Forests. *Expert Systems* (2016).

[11] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7 (2006), 1–30.

[12] Mukund Deshpande and George Karypis. 2004. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177. DOI: http://dx.doi.org/10.1145/963770.963776

[13] Simon Dooms, Toon De Pessemier, and Luc Martens. 2013. MovieTweetings: a Movie Rating Dataset Collected From Twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013.*

[14] Michael Ekstrand and John Riedl. 2012. When Recommenders Fail: Predicting Recommender Failure for Algorithm Selection and Combination. *ACM Conference on Recommender Systems* (2012), 233–236.

[15] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A Free Recommender System Library. In *ACM Conference on Recommender Systems.* 305–308.

[16] Josephine Griffith, Colm O'Riordan, and Humphrey Sorensen. 2012. Investigations into user rating information and predictive accuracy in a collaborative filtering domain. In *ACM Symposium on Applied Computing.* 937–942.

[17] GroupLens. 2016. MovieLens datasets. (2016). http://grouplens.org/datasets/movielens/

[18] A. Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. (1970).

[19] M Hazewinkel. 2008. *Handbook of Algebra.* Number vol. 5 in Handbook of Algebra. Elsevier Science. https://books.google.pt/books?id=fxiZFOXkIKsC

[20] Jonathan L Herlocker, Joseph a. Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* 22, 1 (2004), 5–53. DOI: http://dx.doi.org/10.1145/963770.963772

[21] Balázs Hidasi and Domonkos Tikk. 2012. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In *European Conference on Machine Learning and Knowledge Discovery in Databases.* 67–82.

[22] Zan Huang and Daniel Dajun Zeng. 2011. Why does collaborative filtering work? transaction-based recommendation model validation and selection by analyzing bipartite random graphs. *INFORMS Journal on Computing* 23, 1 (2011), 138–152.

[23] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering. In *ACM Conference on Recommender Systems.* 79–86.

[24] Tamara G Kolda and Brett W Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (2009), 455–500.

[25] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[26] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. 2013. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review* (2013), 1–14.

[27] Christiane Lemke and Bogdan Gabrys. 2010. Meta-learning for time series forecasting and forecast combination. *Neurocomputing* 73, 10-12 (2010), 2006–2016.

[28] Qiang Liu, Shu Wu, and Liang Wang. 2015. COT: Contextual Operating Tensor for Context-Aware Recommender Systems. In *AAAI Conference on Artificial Intelligence.* 203–209.

[29] Pawel Matuszyk and Myra Spiliopoulou. 2014. Predicting the Performance of Collaborative Filtering Algorithms. In *International Conference on Web Intelligence, Mining and Semantics.* 38:1—-38:6.

[30] Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *ACM Conference on Recommender Systems.* 165–172.

[31] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. 2010. Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets. In *ACM Conference on Recommender Systems.* 71–78.

[32] Fábio Pinto, Carlos Soares, and João Mendes-Moreira. 2016. Towards automatic generation of Metafeatures. In *Pacific Asia Conference on Knowledge Discovery and Data Mining.* 215–226.

[33] Ricardo B C Prudêncio and Teresa B Ludermir. 2004. Meta-learning approaches to selecting time series models. *Neurocomputing* 61 (2004), 121–137.

[34] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-thieme. 2011. Fast Context-aware Recommendations with Factorization Machines. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.* 635—-644.

[35] John Rice. 1976. The Algorithm Selection Problem. *Advances in Computers* 15 (1976), 65–118.

[36] André Luis Debiaso Rossi, André Carlos Ponce De Leon Ferreira de Carvalho, Carlos Soares, and Bruno Feres de Souza. 2014. MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data. *Neurocomputing* 127 (2014), 52–64.

[37] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2000. Analysis of Recommendation Algorithms for E-Commerce. In *ACM Conference on Electronic commerce.* 158–167.

[38] Floarea Serban, Joaquin Vanschoren, and Abraham Bernstein. 2013. A survey of intelligent assistants for data analysis. *Comput. Surveys* V, 212 (2013), 1–35.

[39] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver. 2012. TFMAP: Optimizing MAP for Top-N Context-aware Recommendation. In *ACM SIGIR Conference on Research and Development in Information Retrieval.* 155–164.

[40] Carlos Soares. 2015. *labelrank: Predicting Rankings of Labels.* https://cran.r-project.org/package=labelrank

[41] Carlos Soares, Pavel B Brazdil, and Petr Kuba. 2004. A Meta-Learning Method to Select the Kernel Width in Support Vector Regression. *Machine Learning* 54, 3 (2004), 195–209. DOI: http://dx.doi.org/10.1023/B:MACH.0000015879.28004.9b

[42] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. 2008. Tag recommendations based on tensor dimensionality reduction. In *ACM Conference on Recommender Systems.* ACM Press, New York, New York, USA, 43–50. DOI: http://dx.doi.org/10.1145/1454008.1454017

[43] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. 2010. A Unified Framework for Providing Recommendations in Social Tagging Systems Based on Ternary Semantic Analysis. *IEEE Transactions on Knowledge and Data Engineering* 22, 2 (Feb. 2010), 179–192.

[44] H E A Tinsley and S D Brown. 2000. *Handbook of Applied Multivariate Statistics and Mathematical Modeling.* Elsevier Science. https://books.google.pt/books?id=IIbMnrgTpWMC

[45] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311. DOI: http://dx.doi.org/10.1007/BF02289464

[46] Joaquin Vanschoren. 2010. *Understanding machine learning performance with experiment databases.* Ph.D. Dissertation. Katholieke Universiteit Leuven.

[47] Nico Vervliet, Otto Debals, and L ieven De Lathauwer. 2016. Tensorlab 3.0 - Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization. In *Asilomar Conference on Signals, Systems and Computers.* 1733–1738. DOI: http://dx.doi.org/10.1109/ACSSC.2016.7869679

[48] Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent Aspect Rating Analysis Without Aspect Keyword Supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11).* ACM, 618–626.

[49] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. 2010. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *SIAM Conference on Data Mining.* 211–222.

[50] Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. 2014. A survey of collaborative filtering based social recommender systems. *Computer Communications* 41 (2014), 1–10.

[51] Yelp. 2016. Yelp Dataset Challenge. (2016). https://www.yelp.com/dataset_challenge