Modeling Dynamic Interactions over Tensor Streams

Koki Kawabata
SANKEN, Osaka University,, Japan
koki@sanken.osaka-u.ac.jp

Yasuko Matsubara
SANKEN, Osaka University,, Japan
yasuko@sanken.osaka-u.ac.jp

Yasushi Sakurai
SANKEN, Osaka University,, Japan
yasushi@sanken.osaka-u.ac.jp

ABSTRACT

Many web applications, such as search engines and social network services, are continuously producing a huge number of events with a multi-order tensor form, $\{count; query, location, \ldots, timestamp\}$, and so how can we discover important trends to enables us to forecast long-term future events? Can we interpret any relationships between events that determine the trends from multi-aspect perspectives? Real-world online activities can be composed of (1) many time-changing interactions that control trends, for example, competition/cooperation to gain user attention, as well as (2) seasonal patterns that covers trends. To model the shifting trends via interactions, namely *dynamic interactions* over tensor streams, in this paper, we propose a streaming algorithm, DISMO, that we designed to discover **D**ynamic **I**nteractions and **S**easonality in a **M**ulti-**O**rder tensor. Our approach has the following properties. (a) *Interpretable*: it incorporates interpretable non-linear differential equations in tensor factorization so that it can reveal latent interactive relationships and thus generate future events effectively; (b) *Dynamic*: it can be aware of shifting trends by switching multi-aspect factors while summarizing their characteristics incrementally; and (c) *Automatic*: it finds every factor automatically without losing forecasting accuracy. Extensive experiments on real datasets demonstrate that our algorithm extracts interpretable interactions between data attributes, while simultaneously providing improved forecasting accuracy and a great reduction in computational time.

ACM Reference Format:
Koki Kawabata, Yasuko Matsubara, and Yasushi Sakurai. 2023. Modeling Dynamic Interactions over Tensor Streams. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583458

1 INTRODUCTION

Time series forecasting is a powerful tool with a wide range of applications, including user activity modeling [30, 33], recommendation [46, 49], and demand prediction [27, 34]. With the rapid growth of the web-online platforms, it is required for long-term (i.e., multiple steps ahead) forecasting to capture complicated user activities by utilizing rich (i.e., multi-attributed/tensor) data. Online activities are usually non-stationary. Various trends shift streamingly and interactively, but none of their true systems are observable. Furthermore, time-evolving seasonal patterns cover such trends and lead misforecasting. In web-search volume prediction, for example, we want to detect latent user groups interacting with each other, based on query categories, locations, or both. Some of the groups produce growth trends as winners similar to a natural ecosystem. Such relationships change dynamically depending on contextual events (e.g., new product launches) and seasonality (e.g., yearly discount sales). Therefore, we refer to shifting trends with interactions as "*dynamic interactions*" and aim to discover them as a key factor summarizing user preferences and determining trends in streaming data. So, what relationship should we consider for modeling dynamic interactions? How efficiently can we identify dynamic interactions and seasonal patterns simultaneously?

In short, the problem we address is as follows.

Given: a large multi-order tensor stream $\mathcal{X}$, which consists of tuples: $\{count; query, location, \ldots, timestamp\}$,

- **(P1) Find** latent interactions producing trends in $\mathcal{X}$,
- **(P2) Factorize** $\mathcal{X}$ into latent multi-aspect groups, and then
- **(P3) Identify** dynamic interactions and seasonal patterns

in a streaming fashion to continue to forecast long-term future trends accurately. In this paper, we propose a novel tensor factorization technique, namely DISMO[1], that enables the decomposition of **D**ynamic **I**nteractions and **S**easonality in a **M**ulti-**O**rder tensor, as well as a streaming algorithm that updates the model incrementally.

1.1 Preview of the results

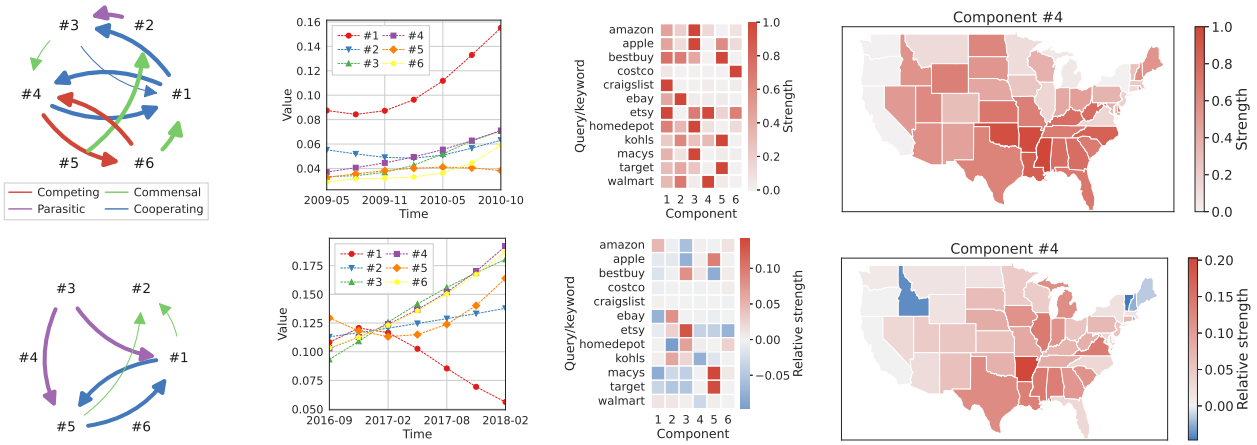
Figure 1 shows an example of streaming DISMO factorization over an E-commerce-related tensor stream, which consists of weekly web-search counts for 12 keywords in 50 states in the US. Our method can automatically discover (a) the latent interaction system, with which it factorizes the tensor into three-aspect factors, i.e., (b) time, (c) query, and (d) location factors.

Concerning (P1), we design the latent interaction system to capture trends underlying tensor streams, as shown in Figure 1 (a), which assumes four important types of interactions. Now it has six latent groups and their relationships are visualized as a graph, where arrow widths indicate the connection intensity. Competition (red arrows) and cooperation (blue arrows) are bidirectional negative/positive interactions, respectively. Commensals (green arrows) and parasites (purple arrows) are one-directional relationships, where the source group benefits from the growth of the destination group. The destination group exhibits no damage to commensals but there are negative effects to parasites. The latent interaction systems are designed with non-linear differential equations, and thus can generate co-evolving trends as shown in Figure 1 (b). For example, the rapid growth of Group #1 can be interpreted as a result of cooperation with Groups #3 and #4; namely it benefits from the growth of the two groups.

[1]Our source code and datasets are publicly available at [2].

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583458



(a) Latent interaction system (b) Time factors (c) Query factors (d) Location factors

Figure 1: Modeling power of the DISMO factorization over an Ecommerce-related tensor stream. Two sets of snapshots taken in February 2011 (top) and May 2018 (bottom) show: (a) latent interactions consisting of four types of interaction relationships; (b) Time components (i.e., latent trends); (c) query components and (d) location components where the top shows component strength and the bottom shows the relative strength variation during the period. DISMO can automatically discover all these factors and thus allow us to interpret the time-varying relationships (i.e., *dynamic interaction*) from multi-aspect perspectives.

To overcome (P2), our model factorizes the original tensor into multi-aspect groups/factors based on the latent interaction system. The top of Figure 1 (c) shows query-related factors, where 12 queries are represented as six latent groups; the larger values show that the queries are more strongly related to corresponding groups. Moreover, location factors can visualize strongly related areas to each latent group as shown in Figure 1 (d).

From the connection between the latent interaction system and multi-aspect factors, we can interpret circumstances on web activities. For example, Craigslist is strongly assigned to Group #1, which has a rapid growth trend. Based on the latent interaction, the growth can be interpreted as a result of cooperation between Groups #3 and #4; namely it benefits from the growth of the two latent groups. On the other hand, Walmart and Costco relate to Groups #4 and #6, which are in a competing relationship as regards their search counts and Group #6 has a more rapid growth trend than Group #4. Similarly, the colored areas in location group #4 show the states where "Walmart" was particularly searched.

Most importantly, our model can realize dynamic interactions for (P3) by changing its factors. The bottom of Figure 1 is another set of multi-aspect factors in a different period from the top, where query/location factors show the relative variation of values in 2011 and 2018 for a comparison of their group allocations. In query factors, Macys moved from Group #3 $\longrightarrow$ #5, which has a rapid growth trend in terms of cooperation. In location factors, our method increases the element of Group #4 for Arkansas due to the growth of their counts.

Consequently, our model satisfies all the requirements for capturing dynamic interactions, i.e., non-linearity for complex trend curves, multi-aspect factors for rich data, and adaptivity for shifting trends, while preserving their interpretability. Also note that all such dynamic interactions are effectively captured by filtering out seasonal patterns. This approach is accurate and scalable to forecast tensor streams as we will describe in Section 6.

1.2 Contributions

In summary, we propose DISMO as an all-in-one algorithm that enables the automatic stream mining of dynamic interactions. Our contributions are summarized as follows.

- *Interpretable*: We formulate a joint model for the non-linear differential equations of interactions and multi-order tensor factorization, which allows us to interpret latent groups in tensors and the relationships behind trends simultaneously.
- *Dynamic*: Our model is designed to extend its factors to adapt to arriving tensor streams incrementally. We show that the mechanism supports not only the understanding of shifting trends with seasonality but also forecasting accuracy.
- *Automatic*: Our algorithm determines the best number of factors to extend based on a lossless data encoding scheme without the need for any user interventions, which makes it easy to analyze large tensor streams.

Outline. The rest of this paper is organized in a conventional way. After introducing related studies in Section 2, we present our proposed model in Section 3 and a route to model optimization in Section 4. Then, we propose both static and streaming optimization algorithms in Section 5. We provide our experimental results in Section 6, followed by a conclusion in Section 7.

2 RELATED WORK

In this section, we briefly describe investigations related to this research. Table 1 summarizes the relative advantages of DISMO with regard to five aspects, where none of the existing methods satisfies the requirements for the efficient mining of dynamic interactions over tensor streams. We separate the details of previous studies into three categories, time series modeling, tensor factorization, and data summarization.

Time series modeling. The state space model (SSM) is a general framework for finding latent dynamics in time series [9, 23].

Table 1: Capabilities of approaches.

	SARIMA [9]	DeepAR [34]	CubeCast [15]	CPD/++ [39]	TRMF [48]	SMF [14]	DISMO
Forecasting	✓	✓	✓		✓	✓	✓
Dynamic interaction							✓
Multi-aspect mining		✓		✓			✓
Online algorithm	some		✓	some		✓	✓
Parameter free			✓				✓

Temporal regularized matrix factorization (TRMF [48]) learns the dimension reduction of matrices with autoregressive regularization for the time dimension but can capture only linear dynamics. Ordinary differential equations (ODE) allow us to obtain deterministic generative models, and thus attract huge interest in the data mining area [10, 28]. In biology, it is well known that the Lotka-Volterra competition (LVC) model [21, 43, 45] effectively captures interactions to produce growing/decaying population dynamics of species, and thus the LVC model has been applied to data mining tasks beyond biological data analysis [24, 42], in which time series data to be forecast are considered as "populations" competing for some common resources. The studies [25, 38] combined non-linear dynamical systems and matrix/tensor factorization to find important relationships between locations. As a more general framework, neural ODEs [6, 8] have also been attracting the attention of many researchers. However, none of these non-linear models still focus on the scalable and stable stream mining. Deep neural network (DNN) models [19, 34, 44, 47] provided alternative solutions that can learn high-dimensional dependencies in time series although the models still suffer in terms of their interpretability.

Data stream mining. Stream/online algorithms provide machine learning solutions in an efficient manner [22]. They are based on one-path data processing and a model update [31, 40], which are applicable to high volume data, and therefore they have proved more significant to the data mining and database community in the last few decades [3–5]. As rich contextual data have become available, multi-aspect mining has posed a more challenging problem [50]. [39] is the first contribution aimed at tensor completion that considers the multi-way evolution of tensors but it cannot handle dynamics that can produce future trends. Matrix/tensor factorization approaches provide component-baseed forecasting, for example, SMF [14], which is online update schemes in real-time forecasting of time series while detecting time-varying seasonalities. However, the component-based forecasting suffers from complex trends that evolves non-linearly. Although CubeCast [15] is a real-time forecasting method for tensor streams, it has no interpretability of dynamic interactions and cannot be appplied to higher-order tensors.

Summarization and clustering. Many data summarization techniques has been applied to obtain meaningful patterns/rules by converting large data to succinct representations [7, 20, 41]. The minimum description length (MDL) principle helps us to obtain concise patterns from data automatically [18, 32, 35, 37]. A recent study formulated MoSSo [16], a streaming lossless data compression technique, although it is designed for dynamic graphs. In previous studies [12, 13], time series are divided into segments with multiple distinct patterns, and the number of patterns is determined based on the MDL, but there has been no focus on stream mining.

Consequently, none of these studies has tried to capture dynamic interactions in tensor streams based on an interpretable, automatic, and scalable approach that we focus on.

3 PROPOSED MODEL

In this section, we describe the tensor streams that we want to analyze and define the formal problem of stream forecasting, and then present our model.

3.1 Problem formulation

We consider a data stream to be a series of non-negative M-order tensors, i.e., $\mathcal{X}(1), \ldots, \mathcal{X}(t), \ldots, \mathcal{X}(T)$, where $\mathcal{X}(t) \in \mathbb{N}^{N_1 \times \cdots \times N_M}$ shows that a tensor has arrived at a time step t, and T increases for each time step. N_m denotes the number of attributes in the m-th mode of $\mathcal{X}(t)$, i.e., $m \in \{1, \ldots, M\}$. Since an unlimited number of tensors arrive, we must consider retaining a part of streams that is much smaller than all the streams for continuous analysis and forecasts. Therefore, letting $\mathcal{X}$ be a set of the most recent ℓ tensors, i.e., $\mathcal{X} \in \mathbb{N}^{N_1 \times \cdots \times N_M \times \ell}$ where $\ell \ll T$, the goal we wish to realize is defined as follows.

Problem 1 (Stream forecasting). ***Given*** *the most recent ℓ-long tensor $\mathcal{X} = \{\mathcal{X}(T-\ell+1), \ldots, \mathcal{X}(T)\}$,* ***Forecast*** *$\ell_s$-steps ahead tensor $\mathcal{X}(T+\ell_s)$ continuously.*

Note that we consistently let the last, i.e., the $(M+1)$-th mode of $\mathcal{X}$, correspond to a time/temporal mode, whereas the other modes are non-temporal modes. In a 3-order tensor $\mathcal{X} \in \mathbb{N}^{N_1 \times N_2 \times \ell}$ of web-search counts, for example, an element x_{ijt} is specified by a query index $i \in \{1, \ldots, N_1\}$, a location index $j \in \{1, \ldots, N_2\}$, and a time point $t \in \{1, \ldots, \ell\}$. Ideal models should reduce more forecasting errors by not only capturing robust long-term trends but also by adapting themselves to new data as quickly as possible if important trends shift. As discussed in the introduction, we focus on modeling long-term trends based on dynamic interactions in an interpretable and scalable manner.

3.2 Proposed solution: DISMO

Here, we present our model, namely DISMO, for the composition of **D**ynamic **I**nteractions and **S**easonality in **M**ulti-**O**rder tensor streams. The model is designed to satisfy the following three requirements for stream forecasting, i.e., Problem 1.

- (P1) interpretable non-linear modeling of interactions,
- (P2) interaction-aware multi-aspect mining, and
- (P3) stream mining of dynamic interactions.

Figure 2 shows an overview of our DISMO model for a tensor $\mathcal{X}$. The remaining subsections provide the building blocks needed to complete the full parameter set for DISMO in detail.

3.2.1 Non-linear modeling of interactions (P1). We first introduce the main concept behind our model to capture dynamic interactions. In theoretical biology, it is well known that interactions between species can be modeled as non-linear behavior, especially if we use the Lotka-Volterra competition model [11, 28]. Assuming here $x_i \geq 0$ denotes the population of the i-th species at a specific time

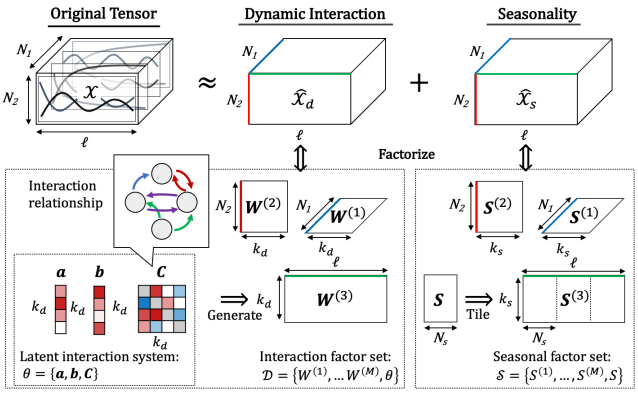


Figure 2: An overview of the DISMO model: an ℓ-long tensor $\mathcal{X}$ is approximated with two parts, dynamic interaction $\hat{\mathcal{X}}_d$ and seasonality $\hat{\mathcal{X}}_s$. Both tensors are factorized into multi-aspect factors $\mathcal{D}$ and $\mathcal{S}$. The temporal factor $\mathbf{W}^{(3)}$ is further compressed into a latent interaction system θ, that allows us to interpret relationships between latent groups as well as generate future trends. More importantly, we consider dynamic interactions as time-evolving $\hat{\mathcal{X}}_d$, and thus allow $\mathcal{D}$ to be replaced in an appropriate manner.

step, it describes population dynamics interacting with k species as:

$$\frac{dx_i}{dt} = a_i x_i \left(1 - \frac{\sum_{j=1}^{k} c_{ij} x_j}{b_i}\right), \quad (1 \le i \le k). \tag{1}$$

The entire non-linear system thus consists of k non-linear differential equations, parameterized by two vectors $\boldsymbol{a} \in \mathbb{R}_+^k$ and $\mathbf{b} \in \mathbb{R}_+^k$, and a matrix $\mathbf{C} \in \mathbb{R}^{k \times k}$. These parameters are interpreted as:

- a_i: intrinsic growth rate of species i;
- b_i: carrying capacity of species i;
- c_{ij}: intra/inter-species interaction strength from the j-th species to the i-th species.

We propose employing these mechanisms to model user behavior on the web. Intuitively, we consider the population dynamics of species as trends in user attention. A larger a indicates that more users are active on a service, suggesting a growing trend. A larger carrying capacity b indicates stronger growth regulation, which allows the number of active users to increase. Most importantly, it is suggested that such trends are influenced by interactions. We can interpret the relationships from the signs of a pair of elements in $\mathbf{C}$ as follows[2].

- $c_{ij} > 0$ and $c_{ji} > 0$: a *competing* relationship;
- $c_{ij} < 0$ and $c_{ji} < 0$: a *cooperating* relationship;
- $c_{ij} < 0$ and $c_{ji} > 0$: a *parasitic* relationship;
- $c_{ij} < 0$ and $c_{ji} = 0$: a *commensal* relationship,

where $i \neq j$ and $i, j \in \{1, \ldots, k\}$. As shown in Figure 2, these relationships can be visualized using a directed graph; nodes and edges are latent species/users and their relationships, respectively.

[2]Note that we fix all the diagonal elements of the interaction coefficient matrix at 1.0, i.e., $c_{ii} = 1.0$ for $i = 1, \ldots, k$, which simplifies the intra-interaction terms.

Unlike identifying real physical systems in species, large tensors contain noise and redundant information when all their attributes are used individually. It is also difficult to identify non-linear dynamical systems, especially as regards high-dimensional and multi-aspect sequences; there are $(N_1 N_2 \cdots N_M)^2$-pair species interactions to estimate. To discover robust trends for long-term forecasting, we assume tensors can be concisely represented by a smaller number of non-linear equations than the number of all attributes when latent user groups are considered as species.

Definition 1 (Latent interaction system: LIS). *Let θ be a parameter set of non-linear equations for modeling a latent interaction system, i.e., $\theta = \{\boldsymbol{a}, \mathbf{b}, \mathbf{C}\}$ for k species/dimensions. Also, let $f_\theta(\theta, w_0, \ell)$ denote the generation of an ℓ-long time series, $\mathbf{W} \in \mathbb{R}^{\ell \times k}$, based on Equation 1 from the initial state $w_0 \in \mathbb{R}^k$ with a given θ.*

3.2.2 Multi-aspect mining for latent interactions (P2). The next question is how to factorize a given tensor based on the latent interaction system θ. To obtain well-summarized and interpretable representations of tensors, we want to assign attributes (e.g., locations and keywords) to latent k groups from each perspective. The assignment should also be independent of seasonality because it typically misleads longer-term ahead forecasting. Motivated by these observations, we consider approximating the original tensor $\mathcal{X}$ with the sum of the two tensors: the dynamic interaction: $\hat{\mathcal{X}}_d$ and the seasonality: $\hat{\mathcal{X}}_s$, and applying multi-linear representations to the tensors individually as shown in Figure 2.

Suppose $\hat{\mathcal{X}}_d \in \mathbb{R}^{N_1 \times \cdots \times N_M \times \ell}$ is an interaction tensor in $\mathcal{X}$. We assume that it has k_d latent factors for each mode, and factorize it using $M+1$ matrices, $\mathbf{W}^{(m)} \in \mathbb{R}^{N_m \times k_d}$ for $m = 1, \ldots, M$ and $\mathbf{W}^{(M+1)} \in \mathbb{R}^{\ell \times k_d}$, so that $\hat{\mathcal{X}}_d$ is computed by:

$$\hat{\mathcal{X}}_d = \sum_{i=1}^{k_d} \mathbf{w}_i^{(1)} \circ \cdots \circ \mathbf{w}_i^{(M)} \circ \mathbf{w}_i^{(M+1)}, \tag{2}$$

where $\circ$ is the outer product. $\mathbf{w}_i^{(m)}$ is the i-th column vector in the matrix $\mathbf{W}^{(m)}$ for each $m = 1, \ldots, M$, but we assume that vectors $\mathbf{w}_i^{(M+1)} \in \mathbf{W}^{(M+1)}$ are sequentially generated by θ based on Equation 1. We keep all the elements in $\mathbf{W}$ nonnegative to preserve the interpretability of the latent interactions. The temporal matrix shows latent k_d dynamic interactions, and non-temporal matrices show the relation strengths to the dynamics for attributes in each mode. This M-way representation forces similar sequences to be summarized into a single latent space. The parameter set for the trend tensor is defined as follows.

Definition 2 (Interaction factor set: $\mathcal{D}$). *Let $\mathcal{D}$ be a set of M nonnegative matrices and an LIS θ, i.e, $\mathcal{D} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M)}, \theta\}$, which represents a dynamic interaction in an ℓ-long tensor $\mathcal{X}$, and we refer to Equation 2 as $f_{\hat{\mathcal{X}}_d}(\mathcal{D}, w_0, \ell)$ to compute an ℓ-long $\hat{\mathcal{X}}_d$, with an interaction factor set $\mathcal{D}$ through Equation 1.*

Suppose $\hat{\mathcal{X}}_s \in \mathbb{R}^{N_1 \times \cdots \times N_M \times \ell}$ is the seasonality tensor for $\mathcal{X}$. We next factorize the tensor into $(M+1)$ matrices to obtain a compact representation of seasonality. Since the last mode of a tensor corresponds to the temporal mode, we let $\mathbf{S} \in \mathbb{R}^{N_s \times k_s}$ denote a latent seasonality matrix, where k_s is the number of latent seasonalities, and N_s is its periodicity term. Likewise, the matrices for non-temporal modes are given by $\mathbf{S}^{(m)} \in \mathbb{R}^{N_m \times k_s}$ for $m \in \{1, \ldots, M\}$. To obtain

an ℓ-long $\hat{\mathcal{X}}_s$, we compute:

$$\hat{\mathcal{X}}_s = \sum_{i=1}^{k_s} \mathbf{s}_i^{(1)} \circ \cdots \circ \mathbf{s}_i^{(M)} \circ \mathbf{s}_i^{(M+1)}, \tag{3}$$

where $\mathbf{S}^{(M+1)}$ is obtained by tiling the $(t \bmod N_s)$-th row vectors of $\mathbf{S}$ for $t \in \{1, \ldots, \ell\}$. Note that k_s is independent of k_d, and we allow the seasonal factors to be negative to make it possible to compress both positive and negative effects simultaneously in a single latent seasonal activity.

Definition 3 (Seasonal factor set: $\mathcal{S}$). *Let $\mathcal{S}$ be a set of $M+1$ matrices for seasonal dynamics in a tensor $\mathcal{X}$, namely, $\mathcal{S} = \{\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(M)}, \mathbf{S}\}$, and we refer to Equation 3 as $f_{\hat{\mathcal{X}}_s}(\mathcal{S}, \ell)$, computing an ℓ-long $\hat{\mathcal{X}}_s$.*

3.2.3 Stream mining dynamic interactions (P3). Thus far, we have discussed the two factor sets, $\mathcal{D}$ and $\mathcal{S}$, that can summarize a given $\mathcal{X}$. However, the model represents only a temporal tensor, and is still insufficient to capture dynamic interactions. The final challenge is to propose how to extend the compatibility of $\mathcal{D}$ for all the tensor streams.

In real-world online activities, the relationship between attributes can shift temporarily or permanently as a result of atypical events, such as a new product being released and discount sales. Therefore, we allow $\mathcal{D}$ to extend as regards any of the aspects to make it aware of trend shifts. Modeling new interaction relationships requires another θ. The rearrangement of latent user groups requires $\mathbf{W}$ to be replaced in a corresponding aspect (e.g., locations and keywords). By this independent tracking of $(M+1)$-aspect pattern shifts, all the model parameters can be summarized more concisely. So, we define $(M+1)$-aspect sets of interaction factors as follows.

Definition 4 (Multi-aspect factor set: $\mathcal{W}^{(m)}$). *Let $\mathcal{W}^{(m)}$ be a nonnegative matrix set of the m-th mode, i.e., we have M sets $\mathcal{W}^{(1)}, \ldots, \mathcal{W}^{(M)}$ in total. Each set $\mathcal{W}^{(m)}$ maintains multiple matrices, i.e., $\mathcal{W}^{(m)} = \{\mathbf{W}_1^{(m)}, \mathbf{W}_2^{(m)}, \ldots, \}$.*

Definition 5 (Latent interaction system set: Θ). *Let Θ be a set of latent interaction systems, i.e., $\Theta = \{\theta_1, \theta_2, \ldots, \}$, where each element is $\theta_i = \{\boldsymbol{a}_i, \mathbf{b}_i, \mathbf{C}_i\}$.*

On the other hand, we assume that seasonalities are long-term patterns, i.e., the patterns vary smoothly but not significantly; therefore, we can maintain them by keeping and updating a single seasonal factor set $\mathcal{S}$. Our final goal is to estimate all these time-varying parameters in a streaming setting. The full parameter set that we want to estimate is eventually defined as follows.

Definition 6 (Full parameter set: $\mathcal{F}$). *Let $\mathcal{F}$ be a full parameter set of DISMO, i.e., $\mathcal{F} = \{\mathcal{W}^{(1)}, \ldots, \mathcal{W}^{(M)}, \Theta, \mathcal{S}\}$, which describes all important dynamics (i.e., dynamic interactions and seasonality) in a tensor stream.*

4 AUTOMATIC TENSOR COMPRESSION

In this section, we propose a criterion for choosing the best model structure of DISMO, $\mathcal{F}$, for a given tensor stream $\mathcal{X}$.

Deciding on an appropriate model structure presents users with highly time-consuming and difficult analytics, and pre-defined models become outdated as we observe new tensors. Therefore, we provide a way of obtaining a concise yet reasonable number of factors without any user intervention. We realize automatic mining with the minimum description length (MDL) principle, because it allows us to measure the model complexity of DISMO. Specifically, the total MDL cost of $\mathcal{F}$ for $\mathcal{X}$ is given as the following equation.

$$< \mathcal{X}; \mathcal{F} > = < \mathcal{F} > \; + \; < \mathcal{X} | \mathcal{F} > , \tag{4}$$

where the right term consists of the model description cost of $\mathcal{F}$ and data encoding cost of $\mathcal{X}$ when given $\mathcal{F}$.

The model description cost $< \mathcal{F} >$ is defined as the number of bits, i.e., the size of the memory needed to store the model parameters.

$$< \mathcal{F} > = < \Theta > \; + \; < \mathcal{W}^{(1)} > \; + \cdots + \; < \mathcal{W}^{(M)} > \; + \; < \mathcal{S} > \; .$$

The cost of the LIS set is further decomposed as follows.

$$\begin{aligned}
< \Theta > \; &= \sum_{\theta \in \Theta} < \theta > , \; < \theta > = < \boldsymbol{a} > \; + \; < \mathbf{b} > \; + \; < \mathbf{C} > \; . \\
< \boldsymbol{a} > \; &= |\boldsymbol{a}| \cdot (\log(k_d) + c_F) + \log^*(|\boldsymbol{a}|), \\
< \mathbf{b} > \; &= |\mathbf{b}| \cdot (\log(k_d) + c_F) + \log^*(|\mathbf{b}|), \\
< \mathbf{C} > \; &= |\mathbf{C}| \cdot (2 \cdot \log(k_d) + c_F) + \log^*(|\mathbf{b}|),
\end{aligned}$$

where $|\cdot|$ shows the number of nonzero elements for a given vector/matrix, $\log^*$ is the universal code length for integers, and c_F is the float point cost[3]. $< \Theta >$ is a function of the numbers of latent factors k_d, and thus minimizing the cost encourages its parameter space to be small and sparse. Likewise, the model costs of our multi-aspect factor sets are defined as follows.

$$\begin{aligned}
< \mathcal{W}^{(m)} > \; &= \sum_{\mathbf{W}^{(m)} \in \mathcal{W}^{(m)}} < \mathbf{W}^{(m)} > , \; < \mathcal{S} > = \sum_{\mathbf{S}^{(m)} \in \mathcal{S}} < \mathbf{S}^{(m)} > , \\
< \mathbf{W}^{(m)} > \; &= |\mathbf{W}^{(m)}| (k_d / N_m) (\log(N_m) + \log(k_d) + c_F) + \log^*(|\mathbf{W}^{(m)}|), \\
< \mathbf{S}^{(m)} > \; &= |\mathbf{S}^{(m)}| (k_s / N_m) (\log(N_m + \log(k_s) + c_F) + \log^*(|\mathbf{S}^{(m)}|).
\end{aligned}$$

To compare the costs of multi-linear factors fairly, we rescale the costs by multiplying (k/N), which allows the model costs to be evaluated per $(k \times k)$ space, and enables us to choose the most contributed factor even when the tensor sizes are skewed.

Next, the data encoding cost measures how well the given model compresses the original data. The Huffman coding scheme [32] enables us to encode $\mathcal{X}$ using our model $\mathcal{F}$. It assigns a number of bits to each element in $\mathcal{X}$, which is the negative log-likelihood under a Gaussian distribution with mean μ and variance σ^2, i.e.,

$$< \mathcal{X} | \mathcal{F} > = \sum_{x \in \mathcal{X}} - \log_2 p_{\mu,\sigma}(x - \hat{x}_d - \hat{x}_s).$$

Now, we have completed the total MDL cost of $\mathcal{F}$ for a tensor $\mathcal{X}$, which makes it possible to decide the numbers of latent states k_d and k_s automatically. In the next section, we discuss how to find the best $\mathcal{F}$ that minimizes Equation 4 efficiently.

5 OPTIMIZATION ALGORITHMS

The previous section described our mathematical concepts that enable us to measure a reasonable representation of the dynamic interaction in tensor streams. In this section, we present effective algorithms with which to estimate the full parameter set $\mathcal{F}$ of the DISMO factorization. We first present a static algorithm for the simplest case where $\mathcal{F}$ has a single dynamics, i.e., $\mathcal{D}$ and $\mathcal{S}$, and then, a streaming algorithm to maintain $\mathcal{F}$ incrementally for pattern shifts.

[3] We used $c_F = 32$ bits.

5.1 Solving static DISMO factorization

Here, we propose an optimization algorithm for estimating both sets, i.e., $\mathcal{D}$ and $\mathcal{S}$. Given a tensor $\mathcal{X}$, the goal is to minimize all the reconstruction errors:

$$\min_{\mathcal{D},\mathcal{S}} \|\mathcal{X} - \hat{\mathcal{X}}_d - \hat{\mathcal{X}}_s\|. \tag{5}$$

Recall that $\hat{\mathcal{X}}_d$ and $\hat{\mathcal{X}}_s$ show estimated tensors by using an interaction factor set $\mathcal{D}$ and a seasonal factor set $\mathcal{S}$, respectively. Since there are no exact solutions of the two factor sets, our algorithm adopts an alternating update[4].

Specifically, each aspect of the factors is updated to minimize the objective by filtering out the effect from either $\hat{\mathcal{X}}_d$ or $\hat{\mathcal{X}}_s$. The first step aims to find the best local θ that minimizes residual errors between $\mathcal{X} - \hat{\mathcal{X}}_s$ and $\hat{\mathcal{X}}_d$. To estimate θ efficiently, we project a residual tensor $\mathcal{X} - \hat{\mathcal{X}}_s$ into the k_d-dimensional sequence $\mathbf{W}^{(M+1)}$ with the non-temporal factors so that it can reduce their reconstruction errors. Based on [36], the m-th mode non-negative matrix can be updated by fixing the other matrices as:

$$\begin{aligned}
\mathbf{P}^{(m)} &= \max(0, (\mathbf{X}^{(m)} - \hat{\mathbf{X}}_s^{(m)})(\odot_{i\neq m}^{M+1}\mathbf{W}^{(i)})), \\
\mathbf{Q}^{(m)} &= \max(0, \mathbf{W}_{old}^{(m)}(\otimes_{i\neq m}^{M+1}\mathbf{W}^{(i)\top}\mathbf{W}^{(i)})), \\
\mathbf{W}_{new}^{(m)} &= \mathbf{W}_{old}^{(m)} \otimes \mathbf{P}^{(m)} \oslash \mathbf{Q}^{(m)},
\end{aligned} \tag{6}$$

where, $\otimes$, $\oslash$, and, $\odot$ are the Hadamard product, element-wise division, and the Khatri-Rao product, respectively. The notations $\odot_{i\neq m}^{M+1}$ and $\otimes_{i\neq m}^{M+1}$ indicate the iteration of the operations for each $i = 1, \ldots, M+1$ except for the m-th mode. The element-wise max operation ensures a non-negative constraint on the updated $\mathbf{W}_{new}^{(m)}$. An LIS θ and its initial state for the expected $\mathbf{W}^{(M+1)}$ are then estimated so that it can minimize the following equation[5]:

$$\{\theta, w_0\} \leftarrow \arg\min_{\theta', w_0'} \|\mathbf{W}_{new}^{(M+1)} - f_\theta(\theta', w_0', \ell)\|. \tag{7}$$

The second step estimates M non-temporal factors by solving Equation 6 for each non-temporal aspect to propagate the feature of smooth latent dynamics $\mathbf{W}^{(M+1)}$.

The third step updates $\mathcal{S}$, given a residual tensor $\mathcal{X} - \hat{\mathcal{X}}_d$. Since we assume no non-negativity on $\mathcal{S}$, we can directly solve Equation 5. The objective function can be rewritten in a matrix form as: $\min_{\mathcal{S}} \|\mathbf{X}^{(m)} - \hat{\mathbf{X}}_d^{(m)} - \hat{\mathbf{X}}_s^{(m)}\|$, where, $\hat{\mathbf{X}}_s^{(m)} = \mathbf{S}^{(m)}(\odot_{i\neq m}^{M+1}\mathbf{S}^{(i)})^\top$. The solution thus becomes:

$$\begin{aligned}
\mathbf{S}^{(m)} &= (\mathbf{X}^{(m)} - \hat{\mathbf{X}}_d^{(m)})(\odot_{i\neq m}^{M+1}\mathbf{S}^{(i)})^\dagger \\
&= (\mathbf{X}^{(m)} - \hat{\mathbf{X}}_d^{(m)})(\odot_{i\neq m}^{M+1}\mathbf{S}^{(i)})(\otimes_{i\neq m}^{M+1}\mathbf{S}^{(i)\top}\mathbf{S}^{(i)})^\dagger,
\end{aligned} \tag{8}$$

where † indicates the Moore–Penrose pseudoinverse. The latter expression is obtained with the properties of the Khatri-Rao product and used for computational efficiency [17]. The new latent seasonal dynamics $\mathbf{S}$ is then obtained by taking the average of updated $\mathbf{S}^{(M+1)}$ for the same seasons.

Overall, the static optimization of DISMO iterates the updating of an LIS θ, non-temporal factors $\mathbf{W}^{(m)}$, and seasonal factors $\mathbf{S}^{(m)}$ until convergence. The time complexity of the static DISMO factorization is given as follows.

[4]The entire algorithm is shown in Appendix A

[5]We employ the Levenberg-Marquardt algorithm [26] to reduce the squared errors between the projected latent dynamics and dynamics generated by solving LSODA [29] for a given regime and its initial state. The initial state is fine-tuned whenever the algorithm needs to update any part of the interaction factor set and forecast.

Lemma 5.1. *Assume that $k = \max(k_d, k_s)$, $n^* = N_s \prod_m N_m$ and $n^+ = N_s + \sum_m N_m$. The time complexity of the DISMO factorization is $O(k^2 n^* \ell + k^2(n^+ + \ell))$. See Appendix A for details.*

This lemma shows that our algorithm greatly reduces the estimation in terms of $\mathbf{C}$ because it is defined as $\mathbf{C} \in \mathbb{R}^{k_d \times k_d}$ in the latent space, not $\mathbf{C} \in \mathbb{R}^{n^* \times n^*}$. The next problem is how to detect dynamic interactions by extending $\mathcal{D}$ incrementally.

5.2 Streaming DISMO factorization

We now address the most important problem: the stream mining of dynamic interactions, by proposing a streaming method, DISMO-STREAM. We derive an automated decision process when the algorithm obtains the best set $\mathcal{D}$ for trend shifts in a current $\mathcal{X}$, and incremental update rules for $\mathcal{S}$.

5.2.1 Identifying dynamic interactions. Here, we introduce how to identify dynamic interactions, i.e., $(M+1)$-way trend shifts, when the characteristics of a tensor stream have evolved.

Algorithm 1 shows the procedure for determining the best interaction factor set $\mathcal{D}_{best}$ while observing the most recent tensor $\mathcal{X}$. Letting $\mathcal{D}_{prev}$ be the interaction factor set used in the previous moment, it considers whether or not one of the factors in the set, i.e., $\Phi \in \mathcal{D}_{prev}$, changes. If so, an arriving trend can be modeled with a new factor estimated with $\mathcal{X}$ or one of the known factors in $\mathcal{F}$. Thus, we employ two sub-algorithms as follows:

(1) FactorCreation: compose a candidate interaction factor set $\mathcal{D}'$ by replacing an element $\Phi \in \mathcal{D}_{prev}$ with the new factor Φ' estimated from scratch with $\mathcal{X}$. A new factor can be estimated using static DISMO factorization by skipping the parts for the fixed parameters. When it estimates a new θ, for example, it only performs step 1.

(2) FactorSearch: compose a candidate interaction factor set $\mathcal{D}'$ by retrieving the best factor Φ' from the known factors $\mathcal{F}$ to replace an element $\Phi \in \mathcal{D}_{prev}$. For example, it computes: $\Phi' = \arg\min_{\theta\in\Theta} \|\mathcal{X} - \hat{\mathcal{X}}_d - \hat{\mathcal{X}}_s\|$ with the other fixed parameters, i.e., non-temporal factors and seasonal factors.

The algorithm performs the above two procedure for each aspect to obtain a set of candidate interaction factor sets, $\mathcal{D}_{cand}$. When we consider 3-order tensor analysis, the size of $\mathcal{D}_{cand}$ becomes seven (one previous set, 3-way generated sets, and 3-way retrieved candidates). Finally, it finds the best shift $\mathcal{D}_{best}$ so that it can minimize the total differential cost, given by:

$$\Delta\ <\mathcal{X}; \mathcal{D}, \mathcal{S}> = \Delta\ <\mathcal{D}>\ +\ <\mathcal{X}|\mathcal{D}, \mathcal{S}>, \tag{9}$$

where $\Delta <\cdot>$ gives the model description cost for a newly added factor, i.e., $\mathbf{W}$ or θ, otherwise, the cost is zero. If a new factor is contained in $\mathcal{D}_{best}$, then the algorithm inserts it into $\mathcal{F}$. This algorithm can keep $\mathcal{F}$ concise yet effective to represent a streaming tensor automatically.

5.2.2 Incremental update of seasonal factors. In addition to capturing dynamic interactions, it is effective to update seasonal factors incrementally. Here, suppose $\mathcal{X}$ is a new observation tensor and $\hat{\mathcal{X}}_d$ is the current dynamic interaction in $\mathcal{X}$. We maintain $\mathcal{S}$ to minimize Equation 5. This online optimization problem is identical to that

Algorithm 1 Streaming DISMO factorization($\mathcal{X}, \mathcal{F}$)

Input: Current tensor $\mathcal{X} \in \mathbb{N}^{N_1 \times \cdots \times N_M \times \ell}$
Output: Updated full parameter set $\mathcal{F}'$
1: $\mathcal{D}_{prev} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M)}, \theta\}$; // $\mathcal{D}_{prev} \in \mathcal{F}$: $\mathcal{D}_{best}$ for the previous window
2: $\mathcal{D}_{cand} = \{\mathcal{D}_{prev}\}$; // A set of candidate interaction factor sets for $\mathcal{X}$
3: /* Consider shifting one of the factors in M aspects */
4: **for** $\Phi \in \mathcal{D}_{prev}$ **do**
5: $\Phi' \leftarrow$ FactorCreation $(\mathcal{X}, \mathcal{D}_{prev}, \mathcal{S})$;
6: Replace $\Phi \in \mathcal{D}_{prev}$ with Φ' to obtain $\mathcal{D}'$; $\mathcal{D}_{cand} \leftarrow \mathcal{D}_{cand} \cup \mathcal{D}'$;
7: $\Phi' \leftarrow$ FactorSearch $(\mathcal{X}, \mathcal{F})$;
8: Replace $\Phi \in \mathcal{D}_{prev}$ with Φ' to obtain $\mathcal{D}'$; $\mathcal{D}_{cand} \leftarrow \mathcal{D}_{cand} \cup \mathcal{D}'$;
9: **end for**
10: /* Insert a generated factor or a non-linear model (optional) */
11: $\mathcal{D}_{best} \leftarrow \underset{\mathcal{D} \in \mathcal{D}_{cand}}{\arg\min} \ \Delta \ < \mathcal{X}; \mathcal{D}, \mathcal{S} > ; \mathcal{F}' \leftarrow \mathcal{F} \cup \mathcal{D}_{best}$;
12: /* Incremental update of seasonal factors based on Equation 10 */
13: $\hat{\mathcal{X}}_d \leftarrow f_{\hat{\mathcal{X}}_d}(\mathcal{D}_{best}, w_0, \ell)$; $\mathcal{S}' \leftarrow \underset{\mathcal{S}}{\arg\min} \|\mathcal{X} - \hat{\mathcal{X}}_d - f_{\hat{\mathcal{X}}_s}(\mathcal{S}, \ell)\|$;
14: **return** $\mathcal{F}' = \{\mathcal{W}^{(1)\prime}, \ldots, \mathcal{W}^{(M)\prime}, \Theta', \mathcal{S}'\}$;

described in [50] and obtains the following update rules.

$$\mathbf{P}^{(m)}_{new} \leftarrow \mathbf{P}^{(m)} + (\mathbf{X}^{(m)} - \hat{\mathbf{X}}^{(m)}_d)(\odot^{M+1}_{i \neq m} \mathbf{S}^{(i)}),$$
$$\mathbf{Q}^{(m)}_{new} \leftarrow \mathbf{Q}^{(m)} + (\otimes^{M+1}_{i \neq m} \mathbf{S}^{(i)\mathsf{T}} \mathbf{S}^{(i)}),$$
$$\mathbf{S}^{(m)}_{new} \leftarrow \mathbf{P}^{(m)} (\mathbf{Q}^{(m)})^{\dagger}. \tag{10}$$

Consequently, streaming DISMO factorization retains only $M+1$ pairs of $\mathbf{P}^{(m)}$ and $\mathbf{Q}^{(m)}$, and solves Equation 10 to obtain $\mathcal{S}$ at any time.

Lemma 5.2. *We assume* $r = \max(|\mathcal{W}^{(1)}|, \ldots, |\mathcal{W}^{(M)}|, |\Theta|)$ *to be the maximum number of factors in* $\mathcal{F}$. *Based on Lemma 5.1, the time complexity of streaming DISMO factorization is* $O(Mr(k^2 n^* \ell + k^2(n^+ + \ell)))$ *per time point. See Appendix A for details.*

This theoretical analysis indicates that our proposed algorithm greatly reduces the computational time needed to find dynamic interactions from $O(r^M)$ when we adopt a naive search, i.e., the combination of all $(M+1)$-aspect factors, which can become significantly large if the number of factors in each mode increases. Therefore, our algorithm has the properties we desire for modeling dynamic interactions in tensor streams, which include interpretable non-linear modeling in dynamic multi-aspect tensor factorization.

6 EXPERIMENTAL EVALUATION

In this section, we describe the performance of DISMO using real datasets. The experiments were designed to answer the following questions about DISMO (-STREAM):

Q1. Accuracy: How accurately does it predict future events?
Q2. Scalability: How does it scale in terms of computational time?
Q3. Effectiveness: How well does it extract latent dynamic patterns?

The datasets were obtained from GoogleTrends [1]. Each tensor contains weekly web-search counts collected over 13 years from 2008 to 2020 and related to 4 kinds of query sets. We collected the data for all 50 states of the US, and normalized their counts in the range 0 to 1. We compare our algorithm with the following state-of-the-art models for time series forecasting, including TRMF [48], SMF [14], CubeCast [15], and DeepAR [34]. The details regarding the use of these algorithms are provided in Appendix B.

Q1. Accuracy. Here, we validated the performance of DISMO by comparing its forecasting accuracy. The performance measure is root mean square error (RMSE), which provides good results when the measures are close to zero. Table 2 shows the overall results, where the bold font and underlines show methods providing the best and second best levels of performance, respectively. We compared the RMSE when we varied the forecasting step ℓ_s in 1–3 quarters of a year (i.e., 13–39 steps) while ℓ was 2 years (i.e., 104 steps). Our method greatly improved the RMSE by employing a non-linear dynamical system and its dynamic shifting mechanism. The auto-regression used in TRMF is insufficient for multi-step forecasting. While DeepAR has an excellent generality when modeling time series, it cannot adjust model parameters incrementally, resulting in accuracy comparable to that of SMF, which suggests the importance of online approaches for time-evolving tensor streams. In particular, the performance of our method is comparable to that of SMF in Sweets dataset. This is because it contains relatively constant seasonal patterns among the four data streams. CubeCast uses a more general non-linear model but our model outperforms it by using a concise yet reasonable non-linear equation for dynamic interaction. For a more detailed evaluation of the effectiveness, we prepared a limited version of the proposed method, namely DISMO–naive by disallowing the use of multiple LISs. DISMO significantly reduces the RMSE compared with DISMO–naive, suggesting that dynamic interaction is effective for time-evolving tensors. Overall, our method achieved the adaptive stream forecasting of tensor streams based on a non-linear dynamical system.

Q2. Scalability. We next evaluated the performance of DISMO in terms of computational time. Figure 3 compares the efficiencies of the non-linear approaches for the results we reported in the previous subsection. Our method consistently outperformed its competitors thanks to our incremental update. Although we omitted the linear methods, which are particularly fast at processing data streams (namely, they are quicker than 1.0 ms at any time point), our algorithm is able to capture dynamic interaction, which makes it possible to understand latent non-linear dynamics and forecast future values with higher accuracy.

Q3. Effectiveness. Finally, we describe how effectively our streaming DISMO factorization captures dynamic interactions and forecasts future trends using dynamic interactions over E-commerce tensor streams. Further results are summarized in Appendix B.

Recall that Figure 1 showed dynamic interactions obtained with DISMO. Figure 1 (a) shows graphical representations of the latent interaction matrix $\mathbf{C}$ in the two LISs θ in 2011 and 2018, which allows us to interpret the latent relationships generating the six trends. Figure 1 (b) corresponds to the $\mathbf{W}^{(3)}$ generated by the two LISs. The six latent sequences summarize important trends in $\mathcal{X}$ as its temporal factor. Figure 1 (c) and (d) correspond to non-temporal components $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ for queries and locations. By imposing non-negativity on them, we can still interpret the relation strength between latent trends and multiple attributes.

Our model can effectively represent the original tensor by shifting either $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$ or θ, while preserving its interpretability for interactions. Specifically, Figure 4 shows snapshots corresponding to the two periods, where our method learned the original tensor (gray points), which consists of a total of 600-dimensional sequences, estimated their latent interactions/seasonalities simultaneously (blue lines), and then generated 13-steps (i.e., a quarter of a year) ahead values (red lines). For example, the trend of Craigslist in 2018 is different from that in 2011 although Craigslist continued

Table 2: Forecasting performance comparison. DISMO outperformed its competitors in terms of RMSE.

data	ℓ_s	DISMO	DISMO-naive	CubeCast	DeepAR	SMF	TRMF
Ecommerce	13	**0.0316 ± 0.0081**	0.0881 ± 0.1239	0.0492 ± 0.0339	0.0638 ± 0.0149	0.0591 ± 0.0163	0.1755 ± 0.0207
	26	**0.0368 ± 0.0103**	0.1122 ± 0.1230	0.0455 ± 0.0269	0.0721 ± 0.0159	0.0604 ± 0.0164	0.1758 ± 0.0198
	39	**0.0425 ± 0.0147**	0.1613 ± 0.1420	0.0431 ± 0.0219	0.0776 ± 0.0167	0.0615 ± 0.0166	0.1781 ± 0.0203
Facilities	13	**0.0356 ± 0.0062**	0.0445 ± 0.0076	0.0890 ± 0.0089	0.0593 ± 0.0146	0.0472 ± 0.0115	0.1390 ± 0.0183
	26	**0.0383 ± 0.0108**	0.0458 ± 0.0093	0.0883 ± 0.0119	0.0666 ± 0.0156	0.0471 ± 0.0125	0.1388 ± 0.0161
	39	**0.0406 ± 0.0131**	0.0466 ± 0.0105	0.0865 ± 0.0137	0.0704 ± 0.0155	0.0482 ± 0.0130	0.1381 ± 0.0152
Sweets	13	**0.0276 ± 0.0146**	0.0297 ± 0.0144	0.0422 ± 0.0209	0.0340 ± 0.0167	0.0280 ± 0.0148	0.0823 ± 0.0124
	26	**0.0279 ± 0.0150**	0.0298 ± 0.0146	0.0405 ± 0.0183	0.0357 ± 0.0167	0.0286 ± 0.0151	0.0826 ± 0.0127
	39	0.0283 ± 0.0149	0.0299 ± 0.0146	0.0393 ± 0.0172	0.0371 ± 0.0166	**0.0275 ± 0.0153**	0.0830 ± 0.0128
VoD	13	**0.0293 ± 0.0121**	0.0558 ± 0.0136	0.0479 ± 0.0294	0.1233 ± 0.0438	0.0447 ± 0.0161	0.2297 ± 0.0489
	26	**0.0336 ± 0.0155**	0.0578 ± 0.0145	0.0423 ± 0.0248	0.1433 ± 0.0435	0.0452 ± 0.0158	0.2280 ± 0.0511
	39	0.0384 ± 0.0194	0.0592 ± 0.0150	**0.0380 ± 0.0203**	0.1505 ± 0.0419	0.0450 ± 0.0160	0.2275 ± 0.0604

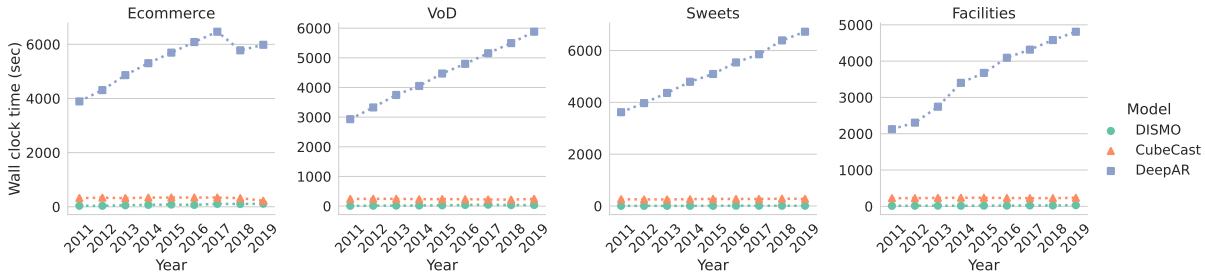


Figure 3: Efficiency of streaming DISMO factorization: the proposed method is always faster than its competitors at any time for summarizing non-linear dynamics and their multi-aspect factors in complex tensor streams.

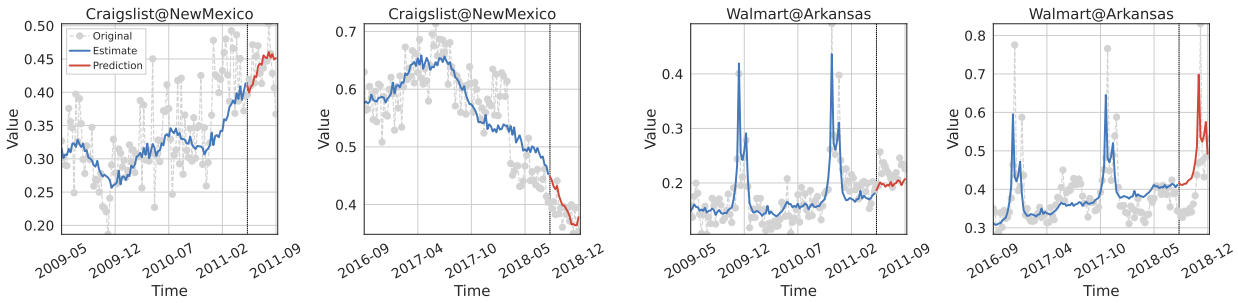


Figure 4: Modeling and forecasting power of DISMO: Our non-linear equations can produce complex curves that can hit multi-steps ahead future events, as well as multiple latent seasonal patterns to uncover the latent interaction more effectively. Detecting dynamic interactions helps to improve forecasting accuracy at any time.

to belong to Component #1. Our method captured this shift by switching to another LIS as shown in Figure 1 (a), resulting in the accurate forecasting of the rapidly decaying count.

Consequently, our non-linear model provides the ability to describe complex curves that can reveal underlying relationships among attributes as LISs, and is suitable for modeling and forecasting tensor streams. By shifting either aspect incrementally, our method keeps its entire model compact, which can allow us to understand the difference between the factors of two periods and also improve forecasting accuracy.

7 CONCLUSION

In this paper, we proposed an efficient and automated streaming method, namely DISMO, for multi-order tensor streams. Our model assumes that tensor streams of web-online activities are composed of two important factors: dynamic interactions and seasonality, and can factorize them into concise representations from multi-aspect perspectives. Our approach has the following properties. *(a) Interpretable:* We employed a non-linear dynamical system for the temporal factor of the dynamic interactions, which allowed us to interpret the relationships between latent groups of attributes and forecast future trends effectively. *(b) Dynamic:* Our proposed algorithm enabled the detection of shifting trends in real time by maintaining multiple multi-aspect factors incrementally. *(c) Automatic:* The numbers of any factors composed by our method are automatically determined based on our lossless data encoding scheme. An experimental evaluation using real tensor streams showed that our proposed method provides superior forecasting accuracy to its competitors, and is more efficient than the state-of-the-art non-linear models.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work was supported by JSPS KAKENHI Grant-in-Aid for Scientific Research Number JP20H00585, JP21H03446, JP22K17896, NICT 03501, MIC/SCOPE JP192107004, JST-AIP JPMJCR21U4, ERCA-Environment Research and Technology Development Fund JPMEERF20201R02.

REFERENCES

[1] GoogleTrends. https://trends.google.com/trends/
[2] Source code and datasets. https://github.com/kokikwbt/dismo
[3] Charu C. Aggarwal. 2006. *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag, Berlin, Heidelberg.
[4] Siddharth Bhatia, Arjit Jain, Shivin Srivastava, Kenji Kawaguchi, and Bryan Hooi. 2022. MemStream: Memory-Based Streaming Anomaly Detection. In *WWW*. 610–621.
[5] TH Hubert Chan, Arnaud Guerqin, and Mauro Sozio. 2018. Fully dynamic k-center clustering. In *WWW*. 579–587.
[6] Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. *NeurIPS* 31 (2018).
[7] Sebastian Dalleiger and Jilles Vreeken. 2022. Discovering Significant Patterns under Sequential False Discovery Control. In *KDD*. 263–272.
[8] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. 2019. Augmented neural odes. *NeurIPS* 32 (2019).
[9] James Durbin and Siem Jan Koopman. 2012. *Time Series Analysis by State Space Methods* (2 ed.). Oxford University Press.
[10] Alan Garfinkel, Jane Shevtsov, and Yina Guo. 2017. *Modeling life: the mathematics of biological systems*. Springer.
[11] James W Haefner. 2005. *Modeling Biological Systems:: Principles and Applications*. Springer Science & Business Media.
[12] Takato Honda, Yasuko Matsubara, Ryo Neyama, Mutsumi Abe, and Yasushi Sakurai. 2019. Multi-aspect mining of complex sensor sequences. In *ICDM*. IEEE, 299–308.
[13] Bryan Hooi, Shenghua Liu, Asim Smailagic, and Christos Faloutsos. 2017. BeatLex: Summarizing and Forecasting Time Series with Patterns. In *ECML/PKDD*. 3–19.
[14] Bryan Hooi, Kijung Shin, Shenghua Liu, and Christos Faloutsos. 2019. *SMF: Drift-Aware Matrix Factorization with Seasonal Patterns*. 621–629.
[15] Koki Kawabata, Yasuko Matsubara, Takato Honda, and Yasushi Sakurai. 2020. Non-Linear Mining of Social Activities in Tensor Streams. In *KDD*. 2093–2102.
[16] Jihoon Ko, Yunbum Kook, and Kijung Shin. 2020. Incremental Lossless Graph Summarization. In *KDD*. 317–327.
[17] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.
[18] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: a partition-and-group framework. In *SIGMOD*. 593–604.
[19] Ming Liang and Xiaolin Hu. 2015. Recurrent convolutional neural network for object recognition. In *CVPR*. 3367–3375.
[20] Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing tensor decomposition for n-ary relational knowledge bases. In *WWW*. 1104–1114.
[21] Alfred James Lotka. 1925. *Elements of physical biology*. Williams & Wilkins.
[22] J Lu, A Liu, F Dong, F Gu, J Gama, and G Zhang. 2019. Learning under Concept Drift: A Review. *TKDE* (2019).
[23] Yasuko Matsubara and Yasushi Sakurai. 2019. Dynamic Modeling and Forecasting of Time-Evolving Data Streams. In *KDD*. 458–468.
[24] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2015. The Web as a Jungle: Non-Linear Dynamical Systems for Co-evolving Online Activities. In *WWW*.
[25] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2016. Non-Linear Mining of Competing Local Activities. In *WWW*.
[26] Jorge J. Moré. 1978. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical Analysis*. 105–116.
[27] Boris N Oreshkin, Grzegorz Dudek, Paweł Pełka, and Ekaterina Turkina. 2021. N-BEATS neural network for mid-term electricity load forecasting. *Applied Energy* 293 (2021), 116918.
[28] Sarah P Otto and Troy Day. 2011. *A biologist's guide to mathematical modeling in ecology and evolution*. Princeton University Press.
[29] Linda Petzold. 1983. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM journal on scientific and statistical computing* 4, 1 (1983), 136–148.
[30] Tobias Preis, Helen Susannah Moat, and H. Eugene Stanley. 2013. Quantifying Trading Behavior in Financial Markets Using Google Trends. *Sci. Rep.* 3 (04 2013).
[31] Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michał Woźniak, and Francisco Herrera. 2017. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* 239 (2017), 39–57.
[32] Jorma Rissanen. 1978. Modeling by shortest data description. *Automatia* 14 (1978), 465–471.
[33] Eduardo J. Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. 2012. Correlating Financial Time Series with Micro-Blogging Activity. In *WSDM*. 513–522.
[34] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.
[35] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. TimeCrunch: Interpretable Dynamic Graph Summarization. In *KDD*. 1055–1064.
[36] Amnon Shashua and Tamir Hazan. 2005. Non-negative tensor factorization with applications to statistics and computer vision. In *ICML*. 792–799.
[37] Mohammad Shokoohi-Yekta, Yanping Chen, Bilson Campana, Bing Hu, Jesin Zakaria, and Eamonn Keogh. 2015. Discovery of meaningful rules in time series. In *KDD*. 1085–1094.
[38] Hyun Ah Song, Bryan Hooi, Marko Jereminov, Amritanshu Pandey, Lawrence T. Pileggi, and Christos Faloutsos. 2017. PowerCast: Mining and Forecasting Power Grid Sequences. In *ECML/PKDD*.
[39] Qingquan Song, Xiao Huang, Hancheng Ge, James Caverlee, and Xia Hu. 2017. Multi-Aspect Streaming Tensor Completion. In *KDD*. 435–443.
[40] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. 2006. Beyond Streams and Graphs: Dynamic Tensor Analysis. In *KDD*. 374–383.
[41] Tsubasa Takahashi, Bryan Hooi, and Christos Faloutsos. 2017. AutoCyclone: Automatic Mining of Cyclic Online Activities with Robust Tensor Factorization. In *WWW*. 213–221.
[42] Bi-Huei Tsai, Chih-Jen Chang, and Chun-Hsien Chang. 2016. Elucidating the consumption and CO2 emissions of fossil fuels and low-carbon energy in the United States using Lotka–Volterra models. *Energy* 100 (2016), 416–424.
[43] Vito Volterra. 1926. Fluctuations in the abundance of a species considered mathematically. *Nature* 118, 2972 (1926), 558–560.
[44] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel wavelet decomposition network for interpretable time series analysis. In *KDD*. 2437–2446.
[45] Peter J Wangersky. 1978. Lotka-Volterra population models. *Annual Review of Ecology and Systematics* 9, 1 (1978), 189–218.
[46] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh V Chawla. 2019. Neural tensor factorization for temporal interaction learning. In *WSDM*. 537–545.
[47] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *WWW*. 351–360.
[48] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. 2016. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In *NeurIPS*. 847–855.
[49] Le Zhang, Tong Xu, Hengshu Zhu, Chuan Qin, Qingxin Meng, Hui Xiong, and Enhong Chen. 2020. Large-Scale Talent Flow Embedding for Company Competitive Analysis. In *WWW*. 2354–2364.
[50] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. 2016. Accelerating Online CP Decompositions for Higher Order Tensors. In *KDD*. 1375–1384.

APPENDIX

A ALGORITHM

In this section, we provide the details of the DISMO optimization algorithm proposed in Subsection 5.1, including its effective initialization algorithm, and then discuss its scalability theoretically.

A.1 Details of static DISMO factorization

Algorithm 2 shows the static optimization process of DISMO. Given a tensor $\mathcal{X}$, and the numbers of latent states k_d and k_s, it reduces the errors between $\mathcal{X}$ and predicted tensors $\hat{\mathcal{X}}_d$ and $\hat{\mathcal{X}}_s$ in an alternative update fashion. After the initialization, it can compute the two reconstruction tensors $\hat{\mathcal{X}}_d$ and $\hat{\mathcal{X}}_s$, each of which is used to reduce the objective squared errors by filtering out the other tensor. Note that the simplest approach to setting k_d, k_s is to search for the best pair of numbers so that it minimizes the total cost, for example, by varying $k_d = 2, 3, \ldots$, and $k_s = 2, 3, \ldots$, simultaneously.

For a stable initialization (in line 2), we first estimate the initial $\mathcal{S}$ by using Equation 8 with $\hat{\mathcal{X}}_d = 0$ set. Then, we compute the residual tensor $\mathcal{X} - \hat{\mathcal{X}}_s$ and only iterate Equation 6 to obtain the initial $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M+1)}$. Finally, we estimate the initial θ over $\mathbf{W}^{(M+1)}$. This initialization effectively avoids divergence due to the initial LIS estimation over a noisy sequence randomly projected by the initial non-temporal factors.

Algorithm 2 Static DISMO factorization($\mathcal{X}, k_d, k_s$)

Input: Tensor $\mathcal{X} \in \mathbb{N}^{N_1 \times \cdots \times N_M \times \ell}$ and the numbers of factors k_d, k_s.
Output: (a) Interaction factor set $\mathcal{D} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M)}, \theta\}$
(b) Seasonal factor set $\mathcal{S} = \{\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(M)}, \mathbf{S}\}$

1: /* Initialization step */
2: $\{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M)}, \theta, \mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(M+1)}\} \leftarrow$ Initialize$(\mathcal{X}, k_d, k_s)$;
3: **repeat**
4: $\quad \hat{\mathcal{X}}_d = f_{\hat{\mathcal{X}}_d}(\mathcal{D}, w_0, \ell)$; // Equation 2
5: $\quad \hat{\mathcal{X}}_s = f_{\hat{\mathcal{X}}_s}(\mathcal{S}, \ell)$; // Equation 3
6: $\quad$ /* Step 1. Estimate the latent interaction system */
7: $\quad \{\theta, w_0\} = \underset{w_0', \theta' \in \mathcal{D}'}{\arg\min} \|\mathcal{X} - \hat{\mathcal{X}}_s - f_{\hat{\mathcal{X}}_d}(\mathcal{D}', w_0', \ell)\|$; //Equation 7
8: $\quad$ /* Step 2. Estimate multi-aspect factors */
9: $\quad$ **for** $m = 1 : M$ **do**
10: $\quad\quad \mathbf{W}^{(m)} \leftarrow \underset{\mathbf{W}'^{(m)} \in \mathcal{D}'}{\arg\min} \|\mathcal{X} - \hat{\mathcal{X}}_s - f_{\hat{\mathcal{X}}_d}(\mathcal{D}', w_0, \ell)\|$; //Equation 6
11: $\quad$ **end for**
12: $\quad$ /* Step 3. Estimate seasonal factors */
13: $\quad$ **for** $m = 1 : M$ **do**
14: $\quad\quad \mathbf{S}^{(m)} \leftarrow \underset{\mathbf{S}'^{(m)} \in \mathcal{S}'}{\arg\min} \|\mathcal{X} - \hat{\mathcal{X}}_d - f_{\hat{\mathcal{X}}_s}(\mathcal{S}', \ell)\|$; //Equation 8
15: $\quad$ **end for**
16: $\quad \mathbf{S} \leftarrow$Mean$(\mathbf{S}^{(M+1)}, N_s)$;
17: **until** convergence;
18: **return** $\{\mathcal{D}, \mathcal{S}\}$;

A.2 Theoretical analysis

We provide the proofs of Lemma 5.1 and Lemma 5.2. Note that $k = \max(k_d, k_s)$, $n^* = N_s \prod_m N_m$, $n^+ = N_s + \sum_m N_m$, and ℓ is the length of $\mathcal{X}$. Lemma 5.1 is given as follows.

Proof. Static DISMO factorization repeatedly computes the dot product with the m-th mode unfolding of $\mathcal{X}$, i.e., $\mathbf{X}^{(m)}$ and the results of the Khatori-Rao product of the matrices $\mathbf{W}^{(m)}$ or $\mathbf{S}^{(m)}$, which requires $O(kn^*\ell)$, with the max operation, requiring $O(kn^+)$. The estimation of an LIS θ iterates the LM algorithm over ℓ-long k_d-dimensional sequences, requiring $O(\#iter \cdot (k_d\ell + k_d^2 + 2k_d))$, but the number of iterations, $\#iter$, is negligible. Thus, the total time complexity of static DISMO factorization is $O(k^2 n^* \ell + k^2(n^+ + \ell))$. □

Here, we also assume $r = \max(|\mathcal{W}^{(1)}|, \ldots, |\mathcal{W}^{(M)}|, |\Theta|)$. Based on the above proof, we obtain Lemma 5.2 as follows.

Proof. Streaming DISMO factorization estimates the two candidate factors for $M + 1$ ways. One is to generate a new factor, which takes $O(k^2 n^* \ell + k^2(n^+ + \ell))$ at most by running Algorithm 2. Another candidate is found by searching for r known factors, and thus the searches need $O(r(k^2 n^* \ell + k^2(n^+ + \ell)))$. Therefore, the total time complexity of streaming DISMO factorization is $O(Mr(k^2 n^* \ell + k^2(n^+ + \ell)))$. □

B EXPERIMENTS

In this section, we describe the experimental setting in detail and provide additional results on real tensor streams with DISMO.

B.1 Experimental setting

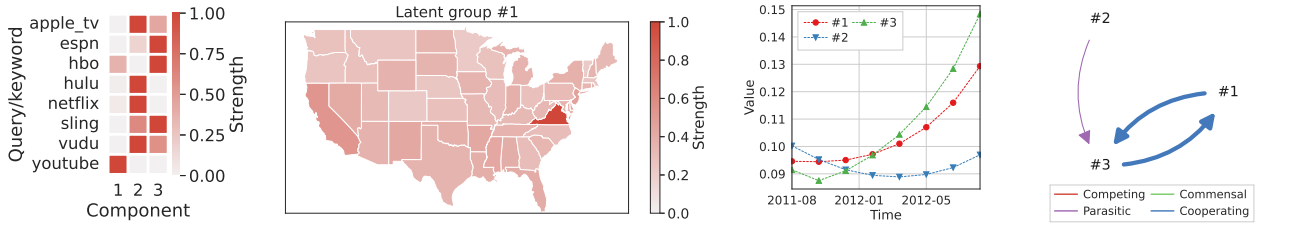
We conducted all our experiments on an Intel Xeon W-2123 3.6GHz quad core CPU with 128GB of memory and running Linux. The details of the baselines we used are summarized as follows.

- TRMF [48]: temporal regularized matrix factorization, which estimates an auto-regression model in its low-rank representation. We searched for the best rank $k \in \{5, 10, 15\}$ and regularization penalties $(\lambda_I, \lambda_{AR}, \lambda_{Lag}) \in \{0.01, 0.1, 1, 10\}$.
- SMF [14]: an online matrix factorization approach that takes seasonal patterns into account. We searched for the best results in its rank $k \in \{5, 10, 15\}$ and learning rate $\alpha \in \{0.1, \ldots, 0.5\}$.
- CubeCast [15]: a stream algorithm that decomposes a tensor into latent non-linear trends and seasonality, and also automatically decomposes their groups of locations based on the two latent dynamics.
- DeepAR [34]: a state-of-the-art neural network for time series forecasting. We built the model with 2-layer 64-unit RNNs, and let it learn for 20 epochs with a learning rate of 0.01. We then chose the best model that produced the lowest validation loss in 10% of training subsequences. The features include the locations, queries, months, and the search counts.

For all the algorithms, including ours, we used the first 3-year tensor to initialize/tune their hyper-parameters. For DISMO, we searched for the best numbers of latent factors by employing a grid

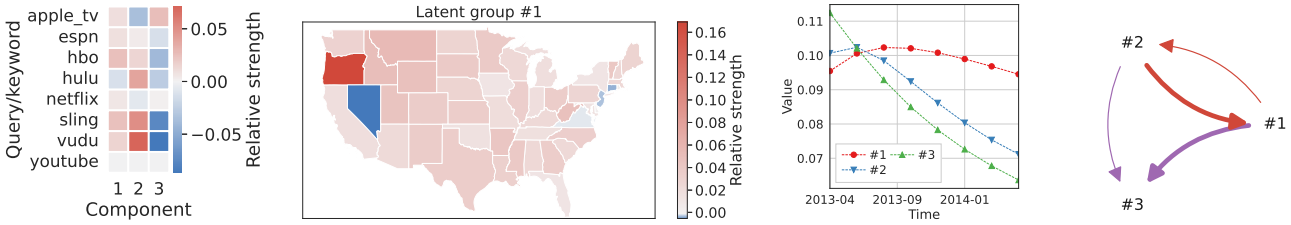
Table 3: GoogleTrends query sets.

Name	Query
Ecommerce	Amazon/Apple/BestBuy/Costco/Craigslist/Ebay/ Etsy/HomeDepot/Kohls/Macys/Target/Walmart
VoD	AppleTV/Disney/ESPN/HBO/Hulu/Netflix/Sling/ YouTube
Facilities	Aquarium/Bookstore/Gym/Library/Museum/ Theater/Zoo
Sweets	Cake/Candy/Chocolate/Cookie/Cupcake/Gum/ Icecream/Pie/Pudding



(a) Query factors (b) Location factors (c) Time factors (d) Latent interaction system

Figure 5: DISMO factorization on VoD-related tensor streams: our method automatically found three latent trends reflecting a rapid growth in interest in VoD in 2012, from which it grouped eight services into: (#1) a representative online video sharing service, (#2) online subscription services, and (#3) television-based on-demand services. The relationship reflects a real-world circumstance where (#2) started attracting users in (#3) without conflicting with (#1). The location factor (#1) also revealed there was a unique behavior on YouTube in the state, Virginia.



(a) Query factors (b) Location factors (c) Time factors (d) Latent interaction system

Figure 6: Dynamic interaction transition from 2012 to 2013. (a) The query factors suggest there was a slight movement (#3→#2), representing the fact that television-based services were adapting to an environment where subscription services were favored. (b) Location factors show that the search count volume in Oregon had grown significantly whereas that for Nevada remained constant. (c) The time factors show that, after the search counts peaked, the three trends became a stable phase and two decay phases, in which (d) the two major VoD groups: (#1) and (#2) competed for users while benefiting from users in (#3).

search in $(k_d, k_s) \in [2, 8]$, so that it minimized Equation 4. For the other models, we performed a leave-one-out cross validation over a set consisting of a two-year tensor for training and a one-year tensor for testing so that they chose their own best hyper-parameters. Note that CubeCast has no parameters to set.

B.2 Effectiveness

Here, we show another result of DISMO. Note that the full sets of queries obtained from GoogleTrends are summarized in Table 3. For each query set, we constructed weekly-basis search counts over the 50 states in the US, and obtained 3-order tensor streams.

Figure 5 shows the results for keywords related to Video-on-Demand services (VoD). From the factor strength in Figure 5 (a), we can interpret the three latent groups as: (#1) YouTube, a representative video sharing service, (#2) online subscription services, and (#3) television-based on-demand services. The three groups reflect the differences in their service types although our method used no prior knowledge. Furthermore, Figure 5 (c) shows three latent dynamics under a rapidly growing phase, and their latent interaction shown in Figure 5 (d) suggests that, although (#1) and (#3) were cooperating to attract more users, some of the users interested in (#3) were moving into (#2) gradually to use fully online services; in the network graph, the source group benefits from the end group. This transition makes sense because (#2) became more favored after approximately one year. In terms of location, Figure 5 (b) shows Virginia paid particular attention to YouTube. This is because the movie, entitled "Virginia", was released in 2012, suggesting that it attracted more users there. According to these observations, our method provides an effective multi-aspect analysis based on latent interactions between users.

It is more interesting to compare dynamic interactions in two distinct periods. Figure 6 summarizes the factors obtained in 2013. Figure 6 (a) and (b) show the relative strength compared from that in Figure 5. While (#3) was in decay, in Figure 6 (a), the strength of (#3) was changing slightly compared with that of (#2), trying to recover their user-friendliness by releasing subscriptions like the original members in (#2), including Netflix and Hulu, which had a better decay trend than (#3). In Figure 6 (c), the three latent trends changed to a stable phase for (#1) and a decaying phase for (#2) and (#3). According to Figure 6 (d), the latent groups (#1) and (#2) were competing for users and benefiting from (#3). That is, YouTube succeeded in continuing to attract users on the search engine as opposed to the subscription services because there were no requirements for users to search for their name on the web after becoming well-known; they have their own applications in several platforms. Regardless of fully-automated mining for dynamic interactions, our method can reveal meaningful pattern shifts in tensor streams.

Consequently, our method can summarize a real-world circumstance as a set of latent interaction-based factors from large tensor streams, which makes it effective for interpreting latent relationships between users as well as forecasting future trends adaptively.