Under consideration for publication in Knowledge and Information Systems

TensorCast: forecasting and mining with coupled tensors

Miguel Araujo[1,2], Pedro Ribeiro[2,3], Hyun Ah Song[1] and Christos Faloutsos[1]

[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA; [2]Computer Science Department, INESC-TEC, Portugal; [3]University of Porto, Portugal

Abstract. Given an heterogeneous social network, can we forecast its future? Can we predict who will start using a given hashtag on twitter? Can we leverage side information, such as who retweets or follows whom, to improve our membership forecasts? We present TensorCast, a novel method that forecasts time-evolving networks more accurately than current state of the art methods by incorporating multiple data sources in coupled tensors. TensorCast is (a) *scalable*, being linearithmic on the number of connections; (b) *effective*, achieving over 20% improved precision on top-1000 forecasts of community members; (c) *general*, being applicable to data sources with different structure. We run our method on multiple real-world networks, including DBLP, epidemiology data, power grid data, and a Twitter temporal network with over *310 million* non-zeros, where we predict the evolution of the activity of the use of political hashtags.

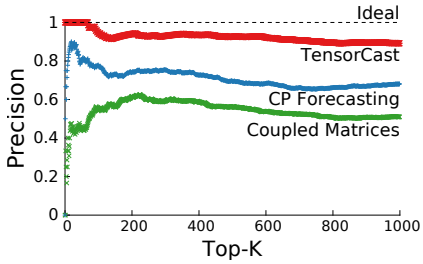
Keywords: time-evolving network; coupled tensor; forecasting;

1. Introduction

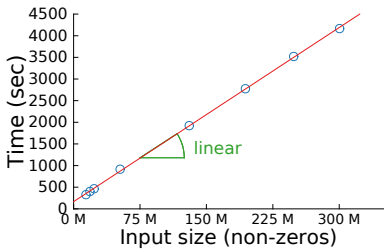
If a group has been discussing the #elections on Twitter, with interest steadily increasing as election day comes, can we predict who is going to join the discussion next week? Intuitively, our forecast should take into account other hashtags (#) that have been used, but also user-user interactions such as followers and retweets.

Similarly, can we predict who is going to publish on a given conference next year? We should be able to make use of, not only the data about where each author previously published, but also co-authorship data and keywords that might indicate a shift in interests and research focus.

Received 04 Jan 2018
Revised 11 Mar 2018
Accepted 14 Apr 2018



(a) **Higher precision** when forecasting $(author, venue)$ relations in the DBLP tensor.



(b) TensorCast **scales linearly** with the number of non-zeros.

Fig. 1. TensorCast is **effective and scalable**.

Today's data sources are often heterogeneous, characterized by different types of entities and relations that we should leverage in order to enrich our datasets. In order to predict the evolution of some of these interactions, we propose to model these heterogeneous graphs as Coupled Tensors that, jointly, generate better predictions than when considered independently.

In particular, we will show how the evolution of user to user connections can be used to forecast user to entity relations, e.g. information about who retweets whom improves the prediction of who is going to use a given hashtag, and co-authorship information improves the prediction of who is going to publish at a given venue.

Informal Problem. ***Forecasting Interactions***

Given historical interaction records between different users and between users and entities.

Find interactions likely to occur in the future efficiently.

Using a *naive* approach, one would have to individually forecast every pair of users and entities - a prohibitively big number that quadratically explodes. How can one avoid quadratic explosion during forecasting? How can we obtain the K likely interactions without iterating through them all?

As a summary of our results, Figure 1a shows that TensorCast is able to achieve 20% more precision than competing methods on the task of predicting who is going to publish on which venue in 2015 using DBLP data. Figure 1b shows TensorCast scaling to hundreds of millions of non-zeros on Twitter data.

We underline our main contributions:

1. **Effectiveness**: TensorCast achieves over 20% higher precision in top-1000 queries and double the precision when finding new relations than comparable alternatives.
2. **Scalability** : TensorCast scales well $(E + N \log N)$ with the input size and is tested in datasets with over *300M* interactions.
3. **Context-awareness**: we show how different data sources can be included in a principled way.

Table 1. Symbols and Definitions

Symbols	Definitions
$\|\boldsymbol{\mathcal{X}}\|_F$	Frobenius norm of tensor $\boldsymbol{\mathcal{X}}$
$\boldsymbol{\mathcal{X}_{(k)}}$	Mode-k matricization
$\boldsymbol{M^t}$	Matrix transpose
$\circ$	Vector outer product
$\odot$	Khatri-rao product
$\otimes$	Hadamard (entrywise) product
$\oslash$	Hadamard (entrywise) division
$\boldsymbol{\mathcal{X}}$	Tensor of our interest (for forecasting)
$\boldsymbol{\mathcal{Y}}$	Coupled tensor
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{T}$	Factor matrices
λ	Parameter for the weight of the coupled factorization
$\boldsymbol{R}$	Set of biggest elements to reconstruct

4. **Tensor Top-K**: we show how to quickly find the K biggest elements of sums of three-way vector outer products under realistic assumptions.

Reproducibility: TensorCast can be obtained at `www.dcc.fc.up.pt/~pribeiro/tensorcast/`.

2. Background

Notation. As common in the literature, we denote vectors by boldface lowercase letters (e.g., $\boldsymbol{a}$), matrices by boldface uppercase letters (e.g., $\boldsymbol{A}$) and tensors by boldface caligraphic letters (e.g., $\boldsymbol{\mathcal{X}}$). For convenience, we refer to the f-th column of $\boldsymbol{A}$ as $\boldsymbol{a_f}$ and to the (i, j, k) entry of 3-mode tensor $\boldsymbol{\mathcal{X}}$ as $\boldsymbol{\mathcal{X}}_{ijk}$. Please refer to Table 1 for operators and additional symbols we use throughout the paper.

2.1. Tensor Factorizations

Tensors are multidimensional arrays that generalize the concept of matrices. As a consequence, they are a popular choice in various applications including representing time-evolving relations, such as Facebook interactions (Papalexakis et al.; 2012), sensor networks (Sun et al.; 2006), EEG data for detecting the origin of epilepsy seizures (Acar et al.; 2007), fMRI analysis (Walker et al.; 2015), image classification (Tao et al.; 2005), or heterogeneous graph analysis (Shi et al.; 2017). When properly applied, tensor factorizations identify the underlying low-dimensional latent structure of the data. The latent factors are then used to identify anomalies, to estimate missing values or to understand how the data was generated in the first place. The PARAFAC (Harshman; 1970) (also called CP)

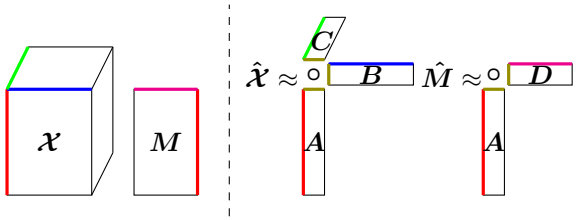


Fig. 2. A simple Coupled Matrix-Tensor Factorization.

decomposition is one of the most popular among the many tensor factorizations flavors (Kolda and Bader; 2009), as it factorizes a tensor into a sum of rank-1 tensors. In three modes, the problem is usually framed as finding factor matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ that minimize the squared error between $\boldsymbol{\mathcal{X}}$ and the reconstructed tensor: $\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \left\| \boldsymbol{\mathcal{X}} - \sum_f \boldsymbol{a_f} \circ \boldsymbol{b_f} \circ \boldsymbol{c_f} \right\|_F^2$.

2.2. Coupled Factorizations

We are often interested in analyzing real-world tensors when additional information is available from distinct sources. For example, in a simple recommendation task with user×movie ratings, we might have user demographics data available which we wish to incorporate when predicting future ratings.

Coupled Matrix-Tensor Factorizations and Coupled Tensor-Tensor Factorizations are a natural extension to the standard tensor factorization formulation. For instance, the factorization of a third-order tensor $\boldsymbol{\mathcal{X}}$ coupled with a matrix $\boldsymbol{M}$ on its first mode can be obtained by minimizing

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}} \left\| \boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}} \right\|_F^2 + \lambda \left\| \boldsymbol{M} - \hat{\boldsymbol{M}} \right\|_F^2 \tag{1}$$

where λ is a parameter representing the strength of the coupling for this task, i.e., how important $\boldsymbol{M}$ is to improve the prediction.

The matrix part of the Coupled Matrix-Tensor Factorization depicted in Figure 2 is useful to model additional static information about one of the modes of the tensor of interest. Whenever the side information available is dynamic (time-evolving), a model where two tensors are coupled along (at least) one of the dimensions is more appropriate, as the time component can be preserved:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{T}} \left\| \boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}} \right\|_F^2 + \lambda \left\| \boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{Y}}} \right\|_F^2 \tag{2}$$

where:

$$\hat{\boldsymbol{\mathcal{X}}} = \sum_f \boldsymbol{a_f} \circ \boldsymbol{b_f} \circ \boldsymbol{t_f}$$

$$\hat{\boldsymbol{\mathcal{Y}}} = \sum_f \boldsymbol{a_f} \circ \boldsymbol{c_f} \circ \boldsymbol{t_f}$$

Many techniques have been proposed to solve this non-negative optimization problem, such as projected Stochastic Gradient Descent (SGD) (Beutel et al.; 2014) (i.e., additive update rules) and multiplicative update rules. Most of this

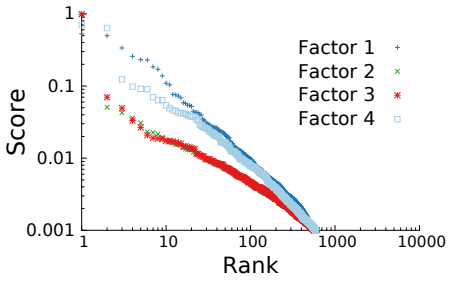


Fig. 3. Scores of the factor vectors are highly skewed. Non-negative factorization of the DBLP `author` × `venue` × `year` tensor. Note the logarithmic scale in both axis.

work extends Lee and Seung's multiplicative matrix updates formulae (Lee and Seung; 2001) for matrices, notably the simple extension for tensors (Welling and Weber; 2001) and the many coupled extensions, e.g. Generalized Tensor Factorization (Yılmaz; 2012; Şimşekli et al.; 2013). Update equations can be found in Appendix 7.

2.3. Skewed building blocks

When factorizing real-life graph data, the scores of the non-negative factors are not uniformly distributed but decrease sharply. For instance, it has been shown that the internal degree distribution of big communities can be well approximated by a power-law across several domains (Araujo et al.; 2014), that eigenvectors of Kronecker graphs exhibit a multinomial distribution (Leskovec et al.; 2005, theorem 3) and multiple generative models where power-law communities arise have been proposed (Pasta et al.; 2013; Zhou et al.; 2008; Xie et al.; 2007). TensorCast leverages this property in order to speed-up its computation of Top-K elements without reconstructing the forecasted *tensor*.

To further strengthen the ubiquity of these structures, Figure 3 shows the scores of 4 factors of the `venue` component of a non-negative factorization of the DBLP `author` × `venue` × `year` tensor we use in the experiments section. Note the skewness of the these scores and that they can be upper-bounded by a power-law.

3. Related Work

3.1. Top-K elements in Matrix Products

Given the widespread applications of matrix factorizations, finding the top-K elements of a matrix product is an important problem with several use cases, from personalized user recommendations to document retrieval.

The problem can be stated as, given matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of sizes $N \times F$ and $M \times F$, respectively, find the top K (i, j) pairs of the $\boldsymbol{AB^t}$ matrix product. Note that the *naive* solution requires $O(NMF)$ operations, iterating over the (originally) implicitly defined reconstruction matrix. Some attention has been given to this problem, since Ram and Gray (Ram and Gray; 2012) proposed the use of Cone Trees to speed-up this search. Other approaches map this problem

into smaller sets of cosine-similarity searches (Teflioudi et al.; 2015), a related but easier problem given the unit-length of the vectors. Approximate methods have also been tried, such as transforming the problem in a near-neighbor search and using locality sensitive hashing (LSH) (Shrivastava and Li; 2014; Neyshabur and Srebro; 2014). However, this is a non-convex optimization problem in general.

3.2. Link Prediction

A large body of literature on link prediction has been created since its introduction (Liben-Nowell and Kleinberg; 2007). In *structural* link prediction, the original problem, the goal is to predict which links are more likely to appear in the future given a current snapshot of the network under analysis. This setting, where it is typical to assume that links are never or seldom removed, has found multiple applications in predicting interactions in protein-protein networks, social networks (e.g., friendship relations) and recommendation problems. The Netflix challenge sprung the creation of several latent factor models with differing structure and/or regularization terms for this task (Koren; 2008; Menon and Elkan; 2011), but there were also several approaches which showed that using the age of the link could lead to improved predictions (Koren; 2010).

On the other hand, given the increased availability of dynamic or time-evolving graphs (frequently used to model evolving relationships between entities over time), *temporal* link prediction methods have been developed to predict future snapshots. In this setting where links are not guaranteed to persist over time, we distinguish methods that rely on collapsing (matricizing) the input data (e.g., exponential decay of edge weights (Sharan and Neville; 2008; Gao et al.; 2011)) from methods that deal directly with the increased dimensionality, such as tensor-based methods. CP Forecasting (Dunlavy et al.; 2011) finds a low-rank PARAFAC factorization and forecasts the time-component in order to incorporate seasonality. TriMine (Matsubara et al.; 2012) similarly factorizes the input tensor, but then applies probabilistic inference in order to identify hidden topics that connect users and entities, which it then draws from in order to generate realistic sequences of future events. These methods are not able to integrate contextual information on their predictions. Other approaches integrate structure and content in the same prediction task, e.g. Gao et al (Gao et al.; 2011) suggest a coupled matrix factorizations and graph regularization technique to obtain the latent factors after an exponential decay of the temporal network.

However, none of these methods fulfills all the requirements for forecasting when contextual information is considered. Table 2 contrasts TensorCast against the state of the art competitors on key specs: (a) linear **scalability** with sparse data; (b) **interpretability** of the underlying model; (c) **time-awareness** for forecasting periodic, growing and/or decaying relations; (d) ability to deal with additional **contextual information**; (e) the ability to **forecast** the disappearance of existing relations; and (f) the ability of providing an **ordered** ranking of future events by likelihood of occurrence.

4. Proposed: TensorCast

We assume a coupled-tensors setting where multiple tensors, possibly with different dimensions, are related by common modes. We will assume that at least

Table 2. TensorCast integrates context and time-awareness.

	Scalability	*Interpretability*	*Coupled setting*	*Time-awareness*	*Context-awareness*	*Forecasting*	*Ordered Forecasting*
Truncated SVD	✓	✓					
Truncated Katz	✓	✓					
Coupled Matrices (e.g., (Gao et al.; 2011))	✓	✓	✓		✓		
VAR (Zellner; 1962), ARIMA (Box and Pierce; 1970), etc.				✓		✓	✓
CP Forecasting (Dunlavy et al.; 2011)	✓	✓		✓		✓	✓
TriMine (Matsubara et al.; 2012)	✓	✓		✓		✓	
TensorCast	✓	✓	✓	✓	✓	✓	✓

one of these tensors is our tensor of interest: it is a 3-dimensional binary tensor and one of the modes corresponds to a time component which we would like to forecast.

There are many scenarios that can be instantiated under this setting: imagine the existence of membership records of the form *(user, topic, time)*, with N unique users and M unique topics (or communities) over T unique time intervals encoded in a $3rd$-order tensor $\boldsymbol{\mathcal{X}} \in \{0,1\}^{N \times M \times T}$. Maybe we also have available an additional collection of user interaction records of the form *(user, user, time)*, similarly encoded in a $3rd$-order tensor $\boldsymbol{\mathcal{Y}} \in \{0,1\}^{N \times N \times T}$. One possible forecasting problem could be framed as predicting which users will interact with which topics in the future, taking advantage of the information from both sources[1].

We are interested in the following general problem:

Problem 1. ***Forecasting Tensor Evolution***
Given *two coupled tensors ($\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$), a number of K relations and S time-steps.*
Forecast*, for the next S time-steps, the ranked list of K likely non-zero elements of $\boldsymbol{\mathcal{X}}$.*

While Problem 1 is interesting by itself, accurate top-K predictions can often be made by identifying which non-zeros constantly appear in the tensor of interest. In the previous example, these would correspond to users that have constantly discussed the same topics over time. Therefore, we define the following related problem:

[1] One of the experiments in Section 5 deals with this scenario.

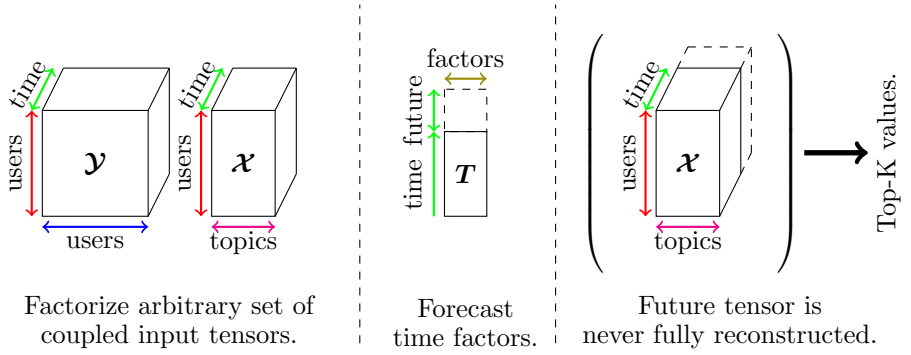


Fig. 4. Overview of TensorCast.

Subproblem 1. ***Forecasting Novel Relations***
Given *two coupled tensors (*$\boldsymbol{\mathcal{X}}$ *and* $\boldsymbol{\mathcal{Y}}$*), a number of* K *relations and* S *time-steps.*
Forecast*, for the next* S *time-steps, the ranked list of* K *likely* ***new*** *relations of* $\boldsymbol{\mathcal{X}}$*.*

We define a **new** or **novel** relation as a non-zero that does not exist in the tensor of interest when the time component is collapsed. We argue that subproblem 1 is more useful in many realistic scenarios where predicting who is joining or leaving a community is more relevant than predicting who is staying. For instance, in the elections example, members who recently joined the discussion are probably easier to influence, while forecasting clients likely to stop doing business with a company is one of the key problems in customer relations.

Overview. TensorCast is comprised of three successive steps, described in more detail in the following subsections:

1. **Non-negative Coupled Factorization:** the factorization will tie together the various input tensors and identify their rank-1 components.
2. **Forecasting:** given the low dimensional space identified, we use standard techniques to forecast the time component.
3. **Top-K elements:** we exploit the factorization structure and identify the top elements without having to reconstruct the prohibitively big future tensor.

Figure 4 illustrates the intuition of our method.

4.1. Non-negative Coupled Factorization

Consider that the tensor of interest, $\boldsymbol{\mathcal{X}}$, is a 3-dimensional $N \times M \times T$ dataset and that the time component corresponds to the last index of the tensor. Then, naively, the number of elements to be forecasted ($S \times N \times M$) is a prohibitive number when we consider $\boldsymbol{\mathcal{X}}$ to be big and sparse.

Therefore, factorizing the input data achieves a two-fold objective: not only does it reduce the number of elements to be forecasted, but perhaps more importantly, it co-clusters similar elements together enabling generalization. A careful factorization will allow the forecast of previously unseen relations. We opted for a non-negative coupled factorization in order to improve the interpretability of

the model; the importance of this feature will be clear when analyzing empirical evidence in Section 5.

We explore how user interactions can be leveraged to improve forecasts of future user-entity relations. Under this assumption, the problem is better modeled as two coupled tensors where tensor $\boldsymbol{\mathcal{Y}}$ is a $N \times N \times T$ symmetric tensor. In order to guarantee convergence, we modify the update of the symmetric factor matrix to

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \otimes \sqrt[3]{\frac{\boldsymbol{\mathcal{X}}_{(1)}(\boldsymbol{B} \odot \boldsymbol{T}) + \lambda \boldsymbol{\mathcal{Y}}_{(1)}(\boldsymbol{A} \odot \boldsymbol{T})}{\boldsymbol{A}(\boldsymbol{B} \odot \boldsymbol{T})^t(\boldsymbol{B} \odot \boldsymbol{T}) + \lambda \boldsymbol{A}(\boldsymbol{A} \odot \boldsymbol{T})^t(\boldsymbol{A} \odot \boldsymbol{T})}}$$

We employ automatic selection of λ, where both tensors are weighted equally by selecting $\lambda = \frac{||\mathcal{X}||_F^2}{||\mathcal{Y}||_F^2}$. The sensitivity analysis on λ is shown later in section 5.

See Appendix 7 for further details.

4.2. Forecasting

Let $\boldsymbol{T}$ be the $T \times F$ factor matrix obtained from the previous step that corresponds to the time component. It consists of a small set of F dense factor vectors, hence easy to forecast, that will provide an approximation $\hat{\boldsymbol{\mathcal{X}}}$ of the next time-step.

The most appropriate forecasting mechanism is data-dependent. We forecast using basic exponential smoothing (Holt's method), but other methods can be applied, e.g. Holt-Winters double exponential smoothing when seasonality is present.

4.3. Tensor Top-K elements

The forecast of the next time-step is a $N \times M \times S$ tensor represented as $\sum_f \boldsymbol{a_f} \circ \boldsymbol{b_f} \circ \boldsymbol{s_f}$ where $\boldsymbol{A}$ is $N \times F$, $\boldsymbol{B}$ is $M \times F$ and $\boldsymbol{S}$ is $S \times F$.

We extend the literature on the retrieval of maximum entries in a matrix product to the tensor case, leveraging the fact that the factorization was not performed on random data but on a graph that follows typical properties. The goal is to identify the K (i, j, k) positions with highest value

$$\sum_f \boldsymbol{A}_{if} \boldsymbol{B}_{jf} \boldsymbol{S}_{kf}$$

We'll start by showing how this could be achieved if the $\hat{\boldsymbol{\mathcal{X}}}$ tensor was rank-1 and how multiple factors can be combined while preserving performance guarantees. We assume that the number of forecasted time-steps is significantly smaller than the number of users or topics (i.e., $S \ll N, M$) and that the number of topics is of the same order of magnitude but smaller than the number of users (i.e., $M < N$).

Top-K of single factor. We start by creating a data structure that lets us obtain the next biggest element in $O(\log(SM))$ time, with only $O(S \log S + M \log M + N \log N + SM)$ preprocessing.

Firstly, we sort the three vectors ($\boldsymbol{s}$, $\boldsymbol{a}$ and $\boldsymbol{b}$) in decreasing order. Note that,

now, not only do we know that the biggest element is given by $\boldsymbol{a}_1\boldsymbol{b}_1\boldsymbol{s}_1$, but also that an element $\boldsymbol{a}_i\boldsymbol{b}_j\boldsymbol{s}_k$ only needs to be considered after $\boldsymbol{a}_{i-1}\boldsymbol{b}_j\boldsymbol{s}_k$, $\boldsymbol{a}_i\boldsymbol{b}_{j-1}\boldsymbol{s}_k$ and $\boldsymbol{a}_i\boldsymbol{b}_j\boldsymbol{s}_{k-1}$ have all been identified as one of the biggest K [2]. Hence, we can create a priority queue which only holds, at most, $O(SM)$ elements at a time.

Combining multiple factors. The major hurdle is handling the interaction between multiple factors. We propose a greedy Top-K selection algorithm that, under realistic scenarios, efficiently achieves this goal. Algorithm 1 illustrates the pseudo code of this procedure.

We keep a list ($\boldsymbol{R}$) of the K biggest positions evaluated so far and $\boldsymbol{F}_i.next$ represents the next element not yet considered in factor's i priority queue, as described in the previous section. In each iteration, we consider the element with the highest score in one of the factors and add it to the list after evaluating it across all the factors. We terminate when the sum of the next best scores on each factor becomes smaller than the K^{th} biggest element in $\boldsymbol{R}$.

input : $\boldsymbol{F}$ - priority queues of factors
input : K - number of elements
output: $\boldsymbol{R}$ - set of biggest elements

while $\sum_i \boldsymbol{F}_i.next.factorScore > \boldsymbol{R}.last.Score$ **do**
  f $\leftarrow \arg\max_i(\boldsymbol{F}_i.next.factorScore)$
  element $\leftarrow \boldsymbol{F}_f.next$
  $\boldsymbol{F}_f.pop$
  $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup element.fullScore$
  if $\boldsymbol{R}.size > K$ **then**
    $\boldsymbol{R} \leftarrow \boldsymbol{R} - \arg\min(\boldsymbol{R})$
  end
end
return $\boldsymbol{R}$

Algorithm 1: TensorCast Top-K Elements

In the following, we prove the correctness and upper bounds on the overall number of elements that need to be evaluated.

Theorem 1. *Algorithm 1 always returns the correct set of Top-k elements.*

Proof. Consider an element x that should be included in $\boldsymbol{R}$ but was never considered. As the algorithm has terminated, it follows that x's score is lower than the sum of all the individual factor scores of elements at the top of each priority queue. However, we know that the smallest element in $\boldsymbol{R}$ is bigger than this, so this is a contradiction and x cannot exist. □

Theorem 1 proves that Algorithm 1 always finds the correct set of elements. We now show that the set of elements that need to be considered is small when factors follow common power-law distributions. We assume $\boldsymbol{a}$ and $\boldsymbol{b}$ follow power-laws of the form $(\alpha-1)x^{-\alpha}$ for $x \geq 1$ and $\alpha > 1$.

Lemma 1. *If factor vectors* **a** *and* **b** *follow power-laws with exponents* α_a *and*

[2] For instance, we know that the second biggest element is one of $\boldsymbol{a}_2\boldsymbol{b}_1\boldsymbol{s}_1$, $\boldsymbol{a}_1\boldsymbol{b}_2\boldsymbol{s}_1$ or $\boldsymbol{a}_1\boldsymbol{b}_1\boldsymbol{s}_2$.

α_b, then a randomly drawn element from any rank-1 frontal slice created as $\boldsymbol{C_{kf}} = \mathbf{a} \circ \mathbf{b}$ asymptotically follows a power-law

$$p_C(z) = (\alpha - 1)z^{-\alpha}$$

where $\alpha = \min(\alpha_a, \alpha_b)$.

Proof. Let X and Y follow power-law distributions of the form

$$p_X(x) = (\alpha_a - 1)x^{-\alpha_a}$$
$$p_Y(y) = (\alpha_b - 1)y^{-\alpha_b}$$

Then $Z = XY$ has probability distribution (Grimmett and Stirzaker; 2001, p. 109):

$$p_Z(z) = \int_1^z p_X(w)p_Y\left(\frac{z}{w}\right)\frac{1}{w}dw = \\ = \frac{(\alpha_a - 1)(\alpha_b - 1)}{\alpha_a - \alpha_b}(z^{-\alpha_a} - z^{-\alpha_b})$$

which tends to a power-law with exponent $-\min(\alpha_a, \alpha_b)$. □

Lemma 1 shows that elements randomly drawn from any rank-1 frontal slice follow a power-law distribution. However, please note that Algorithm 1 iterates over these elements in decreasing order, i.e., deterministically. Therefore, any uncertainty is not related to sampling from the distribution, but rather to the skewness of the factor vectors - how well the power-law assumption holds. Refer back to 2.3 for further details and both theoretical and empirical evidence.

Theorem 2. *Algorithm 1 needs to check at most $KSF^{1+\frac{1}{\alpha}}$ elements if every frontal slice $\mathbf{a_f} \circ \mathbf{b_f}$ follows a power-law.*

Proof. We'll consider the frontal slices one at a time and show that one only needs to check $KF^{1+\frac{1}{\alpha}}$ elements to find the K biggest values of each slice. Let $\alpha_{1..F}$ be the exponents of the power-law of each of the F factor matrices $\mathbf{a_f} \circ \mathbf{b_f}$ of a given frontal slice and let $\alpha_m = \min \boldsymbol{\alpha}$.

The K-th biggest element of $\sum_f \mathbf{a_f} \circ \mathbf{b_f}$ is at least $K^{-\alpha_m}$, as that is the K^{th} biggest value of the slowest decreasing power-law[3]. Given the iterative nature of Algorithm 1, we will prove an upper-bound for the maximum position (i.e., how deep in one of the factors) an element can be, while still having a reconstruction value greater than $K^{-\alpha_m}$. Let x be the position of such element[4], then

$$K^{-\alpha_m} \leq \sum_f x^{-\alpha_f} \leq Fx^{-\alpha_m} \implies x \leq KF^{\frac{1}{\alpha_m}}$$

This means that any top-k element needs to be in a position smaller than $KF^{\frac{1}{\alpha_m}}$ in at least one of the factors, which implies that, in the worst case, Algorithm 1 only needs to check $KF^{\frac{1}{\alpha_m}}F = KF^{1+\frac{1}{\alpha_m}}$ elements to find the K

[3] Remember that $\boldsymbol{A}$ and $\boldsymbol{B}$ are non-negative matrices. In the worst-case, the score of the K^{th} biggest element is taken from a single power-law and the contribution of the rest of the factors is 0, hence $K^{-\alpha_m}$ is a lower-bound for the K^{th} biggest value.

[4] In the worst case scenario, this element is at position x in every of the factors.

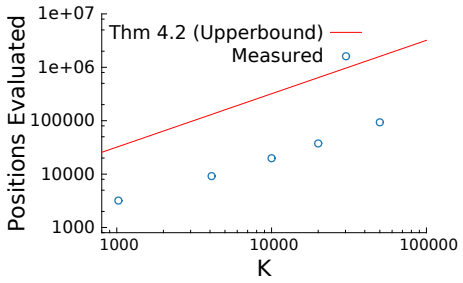


Fig. 5. TensorCast **only checks a linear number of elements** of the tensor.

biggest elements on each frontal slice. Therefore, we can upper-bound the total number of elements checked by $KSF^{1+\frac{1}{\alpha}}$. □

Note that TensorCast is linear on the number of elements we want to obtain times the number of time-steps forecasted. Furthermore, note that this result agrees with intuition: sharper (i.e., quickly decreasing, higher exponent) power-laws require less elements to be checked, while near-clique factors imply lower exponents and more elements to be analyzed.

Figure 5 provides further empirical evidence of the linear growth on the number of values we need to check. We plot the number of positions evaluated as K is increased, on a synthetic network, when forecasting one time-step ($S = 1$), using 8 factors and varying the power-law exponents from 1.5 to 2.2.

4.4. Complexity Analysis.

Observation 1. TensorCast *requires time linear on the number of non-zeros of its input tensors.*

Rationale. TensorCast's time complexity is a sum of its three stages:

1. The coupled-factorization requires linear time on the number of non-zeros.
2. Forecasting is typically linear on the number of timesteps, although it depends on the algorithm selected.
3. As shown in the previous section, identifying the top-K elements is linear on K and sub-quadratic on the number of factors.

5. Experiments

We report experiments to answer the following questions:

Q1. **Scalability**: How fast is TensorCast?

Q2. **Effectiveness** and **Context-awareness**: How does TensorCast's precision compare with its alternatives? How much improvement does contextual data bring?

Q3. **Trend Following**: How capable is TensorCast of detecting and following trends?

Table 3. Summary of real-world networks used.

Users	Groups	Timesteps	Memberships	Interactions	Description
1,734,902	5,476	79	8,049,559	21,423,244	DBLP - venues published and co-authorships.
12,426,133	2,326,843	31	30,281,817	282,280,158	TWITTER - hashtags used and retweets.

Q4. **Precision over Time**: How does TENSORCAST's precision behave as we forecast farther to the future?

Q5. **Sensitivity analysis on** λ: How sensitive is TENSORCAST's performance on the parameter λ?

TENSORCAST is tested on two big datasets detailed in Table 3. In the DBLP dataset, the tensor to be forecasted consists of authors and venues in which they published from 1970 to 2014, while the co-authorship tensor is used as contextual information. Evaluation is performed on the 2015 *author* $\times$ *venue* data. In the TWITTER dataset, the tensor of interest relates users and hashtags (#) they used from June to December 2009, while the auxiliary tensor represents user interactions through re-tweets. Tweets are grouped by week and evaluation is performed on week 51.

Unless otherwise specified, every factorization approach uses 10 factors. On the TWITTER dataset, we weighted the reconstruction of the tensor of interest as 20 times more relevant that the context tensor. On DBLP, we weighted non-zeros of the tensor of interest 2.66 times higher than in the tensor of context (so that both tensors have the same reconstruction error when considering empty factors).

Q1 - Scalability

We start by evaluating our method's scalability when changing the number of non-zeros in the TWITTER dataset[5]. By changing the number of weeks under consideration, we create a sequence of pairs of tensors that increase in size. For each pair, we measure wall-clock time when performing a rank-4 coupled tensor factorization, forecasting and identification of the top-1000 forecasted non-zeros. Figure 1b shows TENSORCAST's linear scalability.

Q2 - Effectiveness and Context-awareness

Figures 1a and 6 showcase TENSORCAST's accuracy on the task of predicting relations on future time steps. While Figure 1a shows TENSORCAST's superior precision as we increase K on the DBLP dataset, Figure 6 focus particularly on forecasting **novel** relations on TWITTER. We would like to highlight the difficulty

[5] We consider the sum of the non-zeros of both tensors.

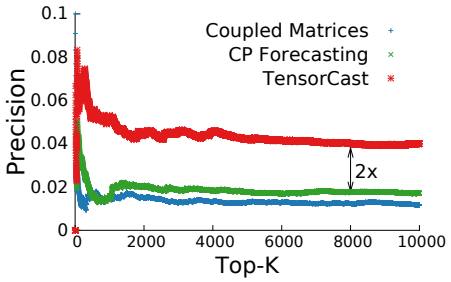


Fig. 6. Double precision when forecasting **novel** $(user, hashtag)$ relations in the TWITTER tensor.

of this task, as we are predicting whether a given user is going to start using a new hashtag on the next week. Nevertheless, TENSORCAST achieves double the precision of competing methods[6].

Furthermore, note the importance of TENSORCAST's ability of being simultaneously contextual and time-aware, as the precision of the current state-of-the-art is limited due to ignoring either one of these aspects.

The competing *CP Forecasting* (Dunlavy et al.; 2011) method was run using Holt forecasting, given the lack of seasonality of the data. The results of the other competitor, *Coupled Matrices*, were obtained by finding non-negative factors that minimize the reconstruction error of the collapsed tensors, weighted for the same importance. For fairness, all appropriate methods use 10 as the number of factors.

Q3 - Trend Following

We evaluate TENSORCAST's ability of predicting an increase or decrease in the activity around a given topic or between a group of users over time. We created a synthetic dataset with 5 hyperbolic communities (i.e., with power-law internal degree distribution) of 100 users over 11 days (10 days are used for the factorization and 1 for evaluation). The average density over the first 10 days equals 15% for all communities, but their density changes differently over time: two communities have their densities increasing at 1% and 2% per day, one has constant density and the other have their density decreasing by 1% and 2% per day.

Figure 7 shows the scores of the 5 columns of the $\boldsymbol{T}$ matrix after factorization, one per line. We can see that linear changes in density correspond to linear changes of the scores and that TENSORCAST correctly forecasts a similar change in the future.

[6] Note that the quality of absolute precision numbers is affected by 1) how imbalanced the two classes are and 2) the cost of false positives. An improvement from 2% to 5% precision might imply that 1 out of 20 phone-calls we make target a potential customer versus every 1 in 50.

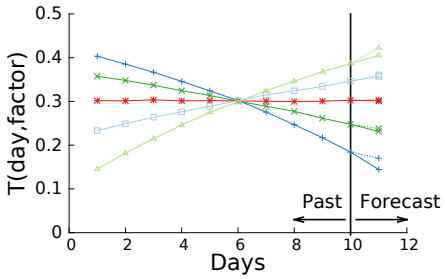


Fig. 7. TensorCast correctly **forecasts growth and decay** of groups in synthetic data. [dashed - forecast; solid - real]

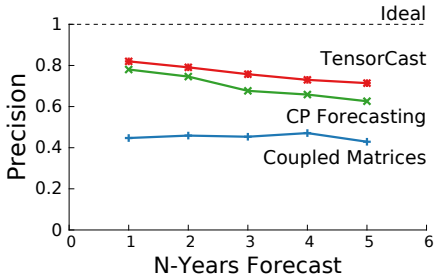


Fig. 8. TensorCast achieves higher precision at every forecasting horizon.

Q4 - Precision over Time

We evaluate TensorCast's precision as the forecasting horizon is increased. We use the DBLP dataset, doing five runs with each method when considering different "training" periods (i.e., the first run considered every publication before 2010, while the last run considered every publication before 2015). For each run, we obtained the 1000 most likely non-zeros for each of the next 5 years and calculated TensorCast's precision. Figure 8 shows, for each method, the average precision for each forecasting horizon.

Q5 - Sensitivity analysis on λ

We analyze TensorCast's sensitivity to the λ parameter, which determines the influence of each tensor. Figure 9 illustrates our methods forecasting performance as λ is changed. The red cross marks the automatic selection defined earlier in section 4. We see that the optimal λ that minimizes the forecasting error is around 0.75, but the automatic choice of $\lambda = 1.12$ is near the optimal point.

6. Discoveries - TensorCast at work

We show that TensorCast is practical and useful on real world dataset by presenting our discoveries:

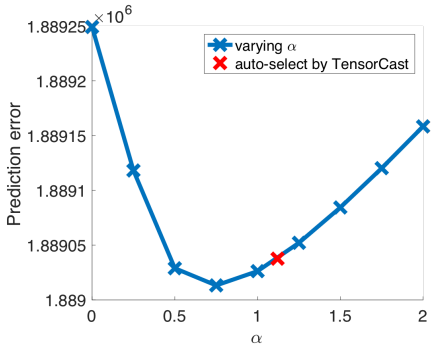


Fig. 9. TensorCast finds near-optimal λ. λ by TensorCast (in *red*) is near optimal

Table 4. Summary of real world datasets with various structures

Name	Description	Structure	Mode	Size	Type
Power grid dataset	Real and imaginary part of the current I_r, I_i (a tensor) and real part of the voltage V_r (a matrix) on CMU campus	A tensor and a matrix	hours of a day; variables (I_r, I_i); days;	24 x 2 x 185 (tensor) 24 x 185 (matrix)	real-valued, full tensor/matrix
Tycho dataset	Epidemics counts of 5 diseases in New York, New Jersey, and Pennsylvania state	three tensors, one for each state	diseases; weeks of a year; years	three tensors of size 52 x 5 x 13	real-valued, full tensor

D1. *Generality*: TensorCast is general and is applicable to various types of real world datasets in different tensor structures.

D2. *Anomaly detection*: the forecast by TensorCast cab be used for anomaly detection.

D3. *Interpretability*: TensorCast provides interpretable results with non-negative factors.

D1 - Generality

In section 5, we have demonstrated TensorCast's behavior on two of the big, sparse, and binary tensors. Can TensorCast be used for different types of real world data that are full, and real-valued coupled tensor-matrix, or three tensors? In this section, we further show the generality of our proposed TensorCast over different types of real world data and varied tensor structures (coupled factorization between a tensor and a matrix, and coupled factorization of three tensors), that is full tensor/matrix. A summary of the additional real world dataset is in Table 4.

1) Power grid dataset: Figure 10 illustrates the power grid dataset we use. These are measurements of the real and imaginary part of the current (I_r and I_i) and the real part of the voltage V_r on the CMU campus from April 29

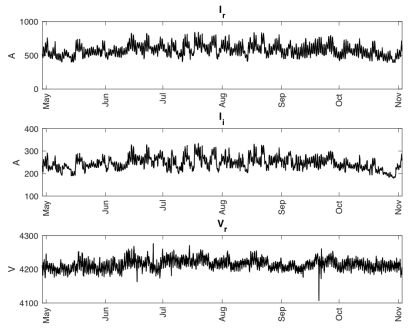


Fig. 10. Power grid dataset

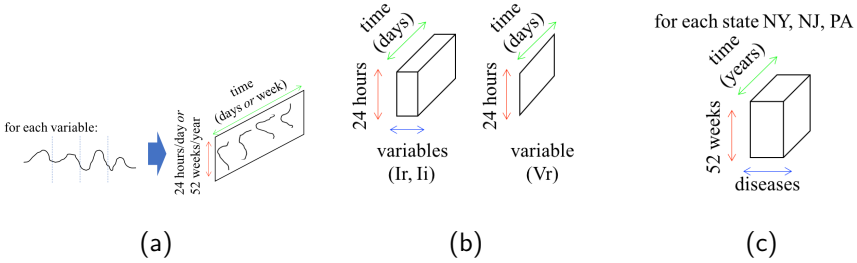


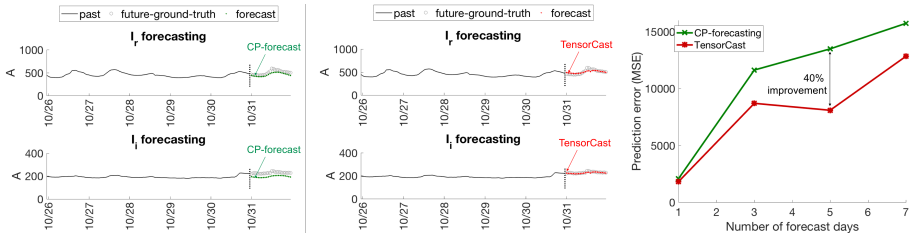
Fig. 11. Tensor/matrix construction. (a) Constructing a tensor/matrix from time-series data. Tensor/matrix structure of (b) power grid dataset, and three tensor structure of (c) Tycho dataset.

2017 to November 2 2017 in hourly interval. Since the voltage is predictable (around $\sim 4.2kV$), we are interested in forecasting the current consumption on campus so that appropriate resources can be provided. Previous work has shown that current consumption is closely related to voltage consumption (in linear relationship) (Song et al.; 2017), so we expect to be able to leverage the voltage consumption pattern over time in order to forecast the current consumption in future.

We can construct a tensor of three modes where each mode corresponds to the days; hours of the day; and I_r and I_i. The coupled matrix is V_r measurements for the days by the hours of the day. How we construct a tensor and a matrix from the time sequences of three variables I_r, I_i, V_r is illustrated in Figure 11 (a) and (b). For each variable I_r, I_i, V_r, we can cut the time sequences into daily units and stack each daily unit one after another to form a slice of a tensor (or a matrix) as shown in (a). Update rules similar to those shown in Appendix 7 can then be formulated.

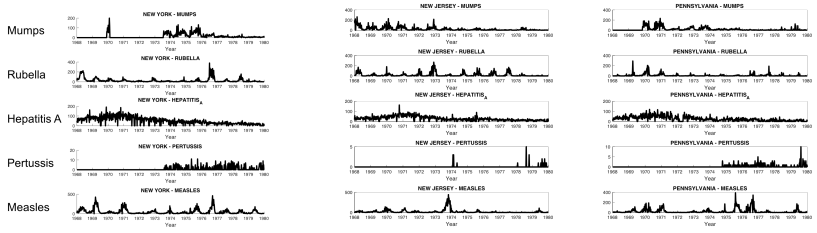
In Figure 12, we show the forecasting result of (a) the competitor algorithm, CP-forecast and of (b) our proposed method TensorCast. We see that while CP-forecasting fails to make accurate forecasts for the next day, making large mistakes, TensorCast forecasts almost realistically.

In Figure 12 (c), we quantitatively evaluate the forecasting accuracy by comparing the mean squared error (MSE) of the forecasting by the competitor al-



(a) CP-forecast (b) TensorCast (c) MSE comparison

Fig. 12. TensorCast forecasts accurately. (a) CP-forecast result. (b) TensorCast forecast result. (c) Quantitative comparison on forecasts.



(a) New York (NY) (b) New Jersey (NJ) (c) Pennsylvania (PA)

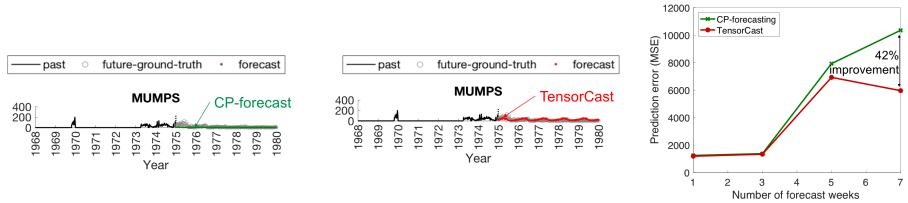
Fig. 13. Tycho dataset, 5 diseases (mumps, rubella, hepatitis A, pertussis, and measles) for 3 states: (a) NY, (b) NJ, and (c) PA

gorithm and that of TensorCast. Results are averaged over 10 trials. We see that, for 1-7 days forecasting, TensorCast is able to achieve lower MSE.

2) Tycho dataset: We also applied our TensorCast to an epidemics dataset [7]. The Tycho project by the University of Pittsburgh gathers the epidemics count (the number of patients) spanning more than 100 years in all states in the USA. In the level 1 type, there are 7 types of disease in total (hepatitis A, measles, mumps, pertussis, polio, rubella, and smallpox) over 51 states. We only use 5 disease types (mumps, rubella, hepatitis A, pertussis, and measles) since they have less missing values. We pick one of the biggest states, New York (NY), and two of the biggest surrounding states, New Jersey (NJ), and Pennsylvania (PA), and forecast the number of occurrences in NY using the contextual information from the nearby states, NJ and PA. We can construct three tensors, one for each state, NY, NJ, and PA, where each mode corresponds to the diseases, weeks of the year, and the years. How we construct three tensors for each state is illustrated in Figure 11 (a) and (c). For Tycho data, variable refers to a disease. As before, update rules can be obtained by simple extensions to the equations in Appendix 7.

Figure 13 illustrates the Tycho dataset of the 5 selected diseases types (mumps, rubella, hepatitis A, pertussis, and measles) in three states, NY, NJ, and PA. We see that some are periodic over the years, while others have a more bursty nature.

[7] https://www.tycho.pitt.edu/



(a) CP-forecast (b) TensorCast (c) MSE comparison

Fig. 14. TensorCast forecasts accurately. (a) CP-forecast result. (b) Quantitative comparison on forecasts. (c) TensorCast forecast result.

We expect that diseases counts of these three states will be somewhat related to each other since they are neighboring states, hence contextual information will enable better forecasts in NYs patients count.

Figure 14 shows the forecasting result from 1975 to 1980 of mumps of (a) the competitor algorithm, CP-forecast, and of (b) our proposed method, TensorCast. We see that, while the competitor algorithm fails to forecast the periodic ups and downs for the next 5 years, TensorCast successfully forecasts this periodicity.

In Figure 14 (c), we quantitatively compare the forecasting result between the competitor algorithm, CP-forecast and TensorCast, for 1-7 years by computing the mean squared error (MSE). As before, we observe that our method consistently achieves lower MSE than the competitor algorithm.

D2 - Anomaly detection

In this section, we show how our method can be extended for anomaly detection tasks. We apply TensorCast on the Tycho and Power grid datasets explained in the previous section and demonstrate its anomaly detection results.

The implicit definition of anomaly is a data point that is unexpected (= surprising = low probability = outlier = rare event). For a cloud of points, an anomaly is a point that is 'too far away' from the rest (Chandola et al.; 2009) - this is exactly what outlier detection methods do (like the time-tested LOF (Breunig et al.; 2000), LOCI (Papadimitriou et al.; 2003), and many more, up to the very recent Random-Cut-Forest-based anomaly detection (Guha et al.; 2016)). Our view point is the same: an entry in the tensor of the form (source, destination, $t+1$), is treated as an anomaly, if it is unexpected (=much different from our forecast), given the past behavior of 'source' and 'destination'. The details of how one can define anomalies are orthogonal to this work - TensorCast just gives expected values for the future, and it is up to the end-user to choose one of the numerous anomaly detection algorithms and thresholds. For ease of presentation, we chose a threshold of 5%, as we explain below.

1) Tycho dataset: In Figure 15, the anomalous time points detected by TensorCast are shown in the red shaded region. These anomalous points are defined as the time points where the difference between the forecasted value and the actually observed value are within top 5% of the deviations observed from the past data. We chose 5% as the threshold for anomaly. Any reasonable value of threshold would be fine (1%, 10%), but we chose 5% because this is the typ-

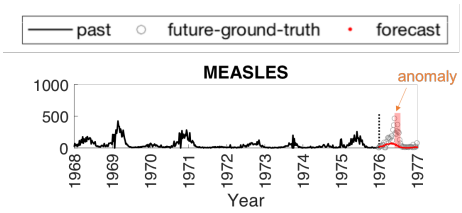


Fig. 15. Anomaly detection in Tycho dataset TensorCast successfully detects sudden increase in measles cases.

ical threshold for determining p-values for statistical-significance analysis. As we can see from the plot, in 1976, there was a sudden increase in the measles count compared to the decreasing trend in the previous years, and TensorCast successfully detected this time point.

What may have caused sudden increase in the measles counts? From the epidemics references, we could find supporting documents that there was a sudden increase in the year 1976 compared to the decreasing trend in the previous years:

"Measles incidence and deaths began to decline in 1965 and continued a 33-year downward trend. This trend was interrupted by epidemics in 1970-1972, ***1976-1978****, and 1989-1991."*[8]

2) Power grid dataset: In Figure 16, anomaly detection on the 2 day forecasting results are shown. Detected anomalous points are indicated in the red shaded region. We observe that there was sudden decrease in I_r (top) and I_i (bottom) compared to the previous steady trend and TensorCast successfully raised a flag for this time period.

Then what happened on 28th and 29th of September that resulted in sudden drop in I_r and I_i in actual observation compared to our forecast? From (Song et al.; 2017), it is observed that I_r and I_i is highly influenced by temperature; higher the temperature, higher the power consumption in I_r and I_i. In the bottom row in Figure 16, temperature of Pittsburgh area is plotted. There was sudden drop in temperature on the 28th of September: the maximum daily temperature dropped from $89°F$ to $70°F$ from 27th to 28th of September. Sudden drop in the temperature resulted in lower power consumption than the forecast/expectation.

A fascinating problem, that TensorCast could help with, is early warnings: This is the 'holy grail' for epilepsy detection (Iasemidis and Sackellares; 1996), with equally high-impact applications in the environment (chaotic analysis for climate change, animal habitats, meteorology (Scheffer et al.; 2009)) and numerous other applications (say, cascade-size forecasting (Cheng et al.; 2014)). Early-warning detection is a huge topic, outside the scope of this paper; we leave it as future research on how to use TensorCast for early warnings.

D3 - Interpretation

In addition to the forecast of new relations, groups found by TensorCast are interpretable due to the non-negativeness of the factors. We highlight two groups we identified on the Twitter dataset with their most used hashtags (#) on

[8] https://www.cdc.gov/mmwr/preview/mmwrhtml/00056803.htm

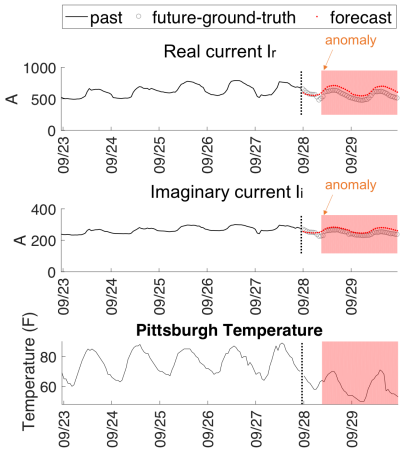


Fig. 16. Anomaly detection in power grid dataset TensorCast successfully detects sudden drop in (top) I_r and (middle) I_i corresponding to sudden temperature drop (bottom).

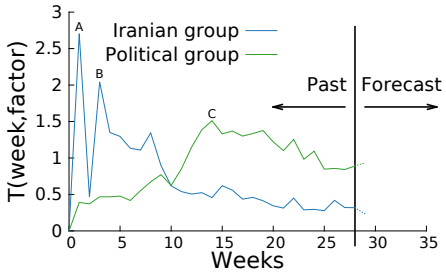


Fig. 17. Increased precision is achieved by grouping interactions. [A-Start of 2009 election protests; B-President Obama references Neda Soltan, killed in the protests; C-Taxpayer March in Washington]

Figure 18 (word size corresponds to importance on factor). The first group corresponds to a group of users who typically use hashtags that show a conservative political orientation: references to the tea party and critics of the healthcare reform. Users in the second group use hashtags related to the Iranian election and human-rights protests, such as #iranelection or #neda, the name of a student who was killed during the protests.

Figure 17 shows TensorCast's ability to predict user interactions based on current interest on a topic. Note that, on the Iranian group, the factorization highlights the week of the elections and the protests (in June), but interest clearly fades in the second-half of the year. On the other hand, we can see that political tags are still used by the same group of users for several months.

Implementation details and Reproducibility. Similarly to its logical steps, TensorCast was implemented as three different modules run in succession:

1. The non-negative coupled factorization was implemented on Matlab using its Tensor Toolbox (Bader et al.; 2015).

(a) **Political group.** Hashtags related to "top conservatives on Twitter" (#tcot) and, respectively, "liberals" (#tlot), #obama, #healthcare (#hcr), "smart girl politics" (#sgp), etc..

(b) **Iranian elections group.** Hashtags related to the Iranian elections and human rights protests.

Fig. 18. TensorCast **finds and forecasts groups with similar interests** on Twitter.

2. Forecasts were done using Gnu's Regression, Econometrics and Time-series Library (GNU gretl) (Baiocchi and Distaso; 2003).
3. Tensor top-K elements' algorithm was implemented in Scala as a stand-alone tool.

TensorCast can be obtained at `www.dcc.fc.up.pt/~pribeiro/tensorcast/`.

7. Conclusions

We presented TensorCast, a method which addresses the forecasting problem on big time-evolving datasets when contextual information is available. We leverage typical graph properties in order to create a linearithmic algorithm that can find novel relations in very big datasets efficiently.

The main advantages of our method are:

1. **Effectiveness**: TensorCast achieves over 20% higher precision in top-1000 queries and double the precision when finding new releations than comparable alternatives.
2. **Scalability** : TensorCast scales linearithmically with the input size and is tested in datasets with over *three hundred million* non-zeros.
3. **Context-awareness**: we show how different data sources can be included in a principled way.
4. **Tensor Top-K**: we show how to quickly find the K biggest elements of sums of three-way vector outer products under realistic assumptions.

Reproducibility: TensorCast can be obtained at `www.dcc.fc.up.pt/~pribeiro/tensorcast/`.

Acknowledgements. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1247489, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. This work was also financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by FCT Fundao para a Cincia e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the Army Research Laboratory, the U.S. Government, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Appendix: Multiplicative Updates of Coupled Tensors Factorization

The non-negative coupled tensor factorization problem

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{T}} \left\| \boldsymbol{\mathcal{X}} - \sum_f \boldsymbol{a_f} \circ \boldsymbol{b_f} \circ \boldsymbol{t_f} \right\|_F^2 + \lambda \left\| \boldsymbol{\mathcal{Y}} - \sum_f \boldsymbol{a_f} \circ \boldsymbol{c_f} \circ \boldsymbol{t_f} \right\|_F^2$$

is well studied and its multiplicative update equations have been previously described in the literature (e.g., considering the dispersion parameter λ (Şimşekli et al.; 2013)). The solution can be found by iteratively updating

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \otimes \frac{\boldsymbol{\mathcal{X}}_{(\boldsymbol{1})}(\boldsymbol{B} \odot \boldsymbol{T}) + \lambda \boldsymbol{\mathcal{Y}}_{(\boldsymbol{1})}(\boldsymbol{C} \odot \boldsymbol{T})}{\boldsymbol{A}(\boldsymbol{B} \odot \boldsymbol{T})^t(\boldsymbol{B} \odot \boldsymbol{T}) + \lambda \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{T})^t(\boldsymbol{C} \odot \boldsymbol{T})}$$

$$\boldsymbol{B} \leftarrow \boldsymbol{B} \otimes \frac{\boldsymbol{\mathcal{X}}_{(\boldsymbol{2})}(\boldsymbol{A} \odot \boldsymbol{T})}{\boldsymbol{B}(\boldsymbol{A} \odot \boldsymbol{T})^t(\boldsymbol{A} \odot \boldsymbol{T})}$$

$$\boldsymbol{C} \leftarrow \boldsymbol{C} \otimes \frac{\boldsymbol{\mathcal{Y}}_{(\boldsymbol{2})}(\boldsymbol{A} \odot \boldsymbol{T})}{\boldsymbol{C}(\boldsymbol{A} \odot \boldsymbol{T})^t(\boldsymbol{A} \odot \boldsymbol{T})}$$

$$\boldsymbol{T} \leftarrow \boldsymbol{T} \otimes \frac{\boldsymbol{\mathcal{X}}_{(\boldsymbol{3})}(\boldsymbol{A} \odot \boldsymbol{B}) + \lambda \boldsymbol{\mathcal{Y}}_{(\boldsymbol{3})}(\boldsymbol{A} \odot \boldsymbol{C})}{\boldsymbol{T}(\boldsymbol{A} \odot \boldsymbol{B})^t(\boldsymbol{A} \odot \boldsymbol{B}) + \lambda \boldsymbol{T}(\boldsymbol{A} \odot \boldsymbol{C})^t(\boldsymbol{A} \odot \boldsymbol{C})}$$

The problem is not as well understood when one of the factorizations is symmetric, e.g., $\hat{\boldsymbol{\mathcal{Y}}} = \sum_f \boldsymbol{a_f} \circ \boldsymbol{a_f} \circ \boldsymbol{t_f}$, as this is no longer a linear problem.

Welling and Weber (Welling and Weber; 2001) note the need for a scaling exponent (for the simple, non-coupled case):

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \otimes \left(\frac{\boldsymbol{\mathcal{X}}_{(\boldsymbol{1})}(\boldsymbol{A} \odot \boldsymbol{T})}{\boldsymbol{A}(\boldsymbol{A} \odot \boldsymbol{T})^t(\boldsymbol{A} \odot \boldsymbol{T})} \right)^{1/d}$$

which should be at least $1/2$ for the matrix case, although no proof is provided. To the best of our knowledge, the best theoretical bound is $1/3$ when the matrix is semi-definite positive (He et al.; 2011). Empirical results (for the coupled case) indicate that removing the exponent ($d = 1$) might eliminate the convergence guarantees, but even small perturbations converge (e.g., 0.98 in (Ermiş et al.; 2013)).

We recommend an exponent of $1/3$, as convergence is exponentially fast in any case.

References

Acar, E., Aykut-Bingol, C., Bingol, H., Bro, R. and Yener, B. (2007). Multiway analysis of epilepsy tensors, *Bioinformatics* **23**(13): i10–i18.

Araujo, M., Günnemann, S., Mateos, G. and Faloutsos, C. (2014). Beyond blocks: Hyperbolic community detection, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 50–65.

Bader, B. W., Kolda, T. G. et al. (2015). Matlab tensor toolbox version 2.6, Available online.
URL: *http://www.sandia.gov/ tgkolda/TensorToolbox/*

Baiocchi, G. and Distaso, W. (2003). Gretl: Econometric software for the gnu generation, *Journal of applied econometrics* **18**(1): 105–110.

Beutel, A., Talukdar, P. P., Kumar, A., Faloutsos, C., Papalexakis, E. E. and Xing, E. P. (2014). Flexifact: Scalable flexible factorization of coupled tensors on hadoop., *SDM*, SIAM, pp. 109–117.

Box, G. E. and Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, *Journal of the American statistical Association* **65**(332): 1509–1526.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. and Sander, J. (2000). Lof: identifying density-based local outliers, *ACM sigmod record*, Vol. 29, ACM, pp. 93–104.

Chandola, V., Banerjee, A. and Kumar, V. (2009). Anomaly detection: A survey, *ACM computing surveys (CSUR)* **41**(3): 15.

Cheng, J., Adamic, L., Dow, P. A., Kleinberg, J. M. and Leskovec, J. (2014). Can cascades be predicted?, *Proceedings of the 23rd international conference on World wide web*, ACM, pp. 925–936.

Dunlavy, D. M., Kolda, T. G. and Acar, E. (2011). Temporal link prediction using matrix and tensor factorizations, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **5**(2): 10.

Ermiş, B., Cemgil, A. T. and Acar, E. (2013). Generalized coupled symmetric tensor factorization for link prediction, *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, IEEE, pp. 1–4.

Gao, S., Denoyer, L. and Gallinari, P. (2011). Temporal link prediction by integrating content and structure information, *Proceedings of the 20th ACM international conference on Information and knowledge management*, ACM, pp. 1169–1174.

Grimmett, G. and Stirzaker, D. (2001). *Probability and random processes*, Oxford university press.

Guha, S., Mishra, N., Roy, G. and Schrijvers, O. (2016). Robust random cut forest based anomaly detection on streams, *International Conference on Machine Learning*, pp. 2712–2721.

Harshman, R. A. (1970). Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis.

He, Z., Xie, S., Zdunek, R., Zhou, G. and Cichocki, A. (2011). Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering, *IEEE Transactions on Neural Networks* **22**(12): 2117–2131.

Iasemidis, L. D. and Sackellares, J. C. (1996). review: Chaos theory and epilepsy, *The Neuroscientist* **2**(2): 118–126.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications, *SIAM review* **51**(3): 455–500.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model, *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 426–434.

Koren, Y. (2010). Collaborative filtering with temporal dynamics, *Communications of the ACM* **53**(4): 89–97.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization, *Advances in neural information processing systems*, pp. 556–562.

Leskovec, J., Chakrabarti, D., Kleinberg, J. and Faloutsos, C. (2005). Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication, *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, pp. 133–145.

Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks, *Journal of the American society for information science and technology* **58**(7): 1019–1031.

Matsubara, Y., Sakurai, Y., Faloutsos, C., Iwata, T. and Yoshikawa, M. (2012). Fast mining and forecasting of complex time-stamped events, *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 271–279.

Menon, A. K. and Elkan, C. (2011). Link prediction via matrix factorization, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 437–452.

Neyshabur, B. and Srebro, N. (2014). On symmetric and asymmetric lshs for inner product search, *arXiv preprint arXiv:1410.5518* .

Papadimitriou, S., Kitagawa, H., Gibbons, P. B. and Faloutsos, C. (2003). Loci: Fast outlier detection using the local correlation integral, *Data Engineering, 2003. Proceedings. 19th International Conference on*, IEEE, pp. 315–326.

Papalexakis, E. E., Faloutsos, C. and Sidiropoulos, N. D. (2012). Parcube: Sparse parallelizable tensor decompositions, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 521–536.

Pasta, M. Q., Jan, Z., Sallaberry, A. and Zaidi, F. (2013). Tunable and growing network generation model with community structures, *Cloud and Green Computing (CGC), 2013 Third International Conference on*, IEEE, pp. 233–240.

Ram, P. and Gray, A. G. (2012). Maximum inner-product search using cone trees, *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 931–939.

Scheffer, M., Bascompte, J., Brock, W. A., Brovkin, V., Carpenter, S. R., Dakos, V., Held, H., Van Nes, E. H., Rietkerk, M. and Sugihara, G. (2009). Early-warning signals for critical transitions, *Nature* **461**(7260): 53.

Sharan, U. and Neville, J. (2008). Temporal-relational classifiers for prediction in evolving domains, *2008 Eighth IEEE International Conference on Data Mining*, IEEE, pp. 540–549.

Shi, C., Li, Y., Zhang, J., Sun, Y. and Philip, S. Y. (2017). A survey of heterogeneous information network analysis, *IEEE Transactions on Knowledge and Data Engineering* **29**(1): 17–37.

Shrivastava, A. and Li, P. (2014). Improved asymmetric locality sensitive

hashing (alsh) for maximum inner product search (mips), *arXiv preprint arXiv:1410.5410* .

Şimşekli, U., Ermiş, B., Cemgil, A. T. and Acar, E. (2013). Optimal weight learning for coupled tensor factorization with mixed divergences, *21st European Signal Processing Conference (EUSIPCO 2013)*, IEEE, pp. 1–5.

Song, H. A., Hooi, B., Jereminov, M., Pandey, A., Pileggi, L. and Faloutsos (2017). Powercast: Mining and forecasting power grid sequences, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. xx–xx.

Sun, J., Tao, D. and Faloutsos, C. (2006). Beyond streams and graphs: dynamic tensor analysis, *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 374–383.

Tao, D., Maybank, S., Hu, W. and Li, X. (2005). Stable third-order tensor representation for color image classification, *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society, pp. 641–644.

Teflioudi, C., Gemulla, R. and Mykytiuk, O. (2015). Lemp: Fast retrieval of large entries in a matrix product, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM, pp. 107–122.

Walker, P. B., Gilpin, S., Fooshee, S. and Davidson, I. (2015). Constrained tensor decomposition via guidance: Increased inter and intra-group reliability in fmri analyses, *International Conference on Augmented Cognition*, Springer, pp. 361–369.

Welling, M. and Weber, M. (2001). Positive tensor factorization, *Pattern Recognition Letters* **22**(12): 1255–1261.

Xie, Z., Li, X. and Wang, X. (2007). A new community-based evolving network model, *Physica A: Statistical Mechanics and its Applications* **384**(2): 725–732.

Yılmaz, Y. K. (2012). *Generalized tensor factorization*, PhD thesis, Citeseer.

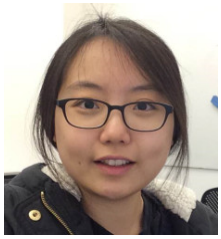
Zellner, A. (1962). An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias, *Journal of the American statistical Association* **57**(298): 348–368.

Zhou, X., Xiang, L. and Xiao-Fan, W. (2008). Weighted evolving networks with self-organized communities, *Communications in Theoretical Physics* **50**(1): 261.

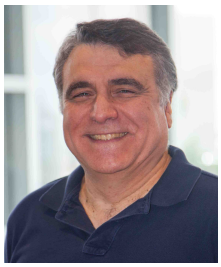
Author Biographies

Miguel Araujo is a Senior Data Scientist at Feedzai working on new machine learning techniques to detect fraud in financial data. He received his Ph. D. in the CMUPortugal dual degree program in Computer Science between Carnegie Mellon University and the University of Porto, and earned his M.Sc. in Informatics and Computing Engineering at the University of Porto where he also worked as a research assistant at the Laboratory of Artificial Intelligent and Decision Support. He has published over 15 papers in reputable venues, received 2 best paper awards (including the best paper of ICDM 2017) and has one patent pending.

Pedro Ribeiro is an Assistant Professor in the Computer Science Department of the School of Sciences at the University of Porto, Portugal. He was awarded his Ph.D. in Computer Science from the University of Porto in 2011. He has a core background on algorithms and data structures and his main research line is focused on network science and on the analysis of complex networks, particularly on the algorithmic aspects of discovering interesting patterns. He also has a strong interest in parallel and distributed computing, computer science education and programming contests.



Hyun Ah Song is a PhD student in the Machine Learning Department at Carnegie Mellon University, USA. Before joining Carnegie Mellon, she worked in Biomedical Department at Korea Institute of Science and Technology (KIST), Korea, where she worked on lipreading system for the senior/disabled. She received her M.S. in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Korea, where she worked on representation learning using neural networks. She received her B.S in Environmental Science and Engineering from Ewha Womans University, Korea. Her current research includes time series analysis in various applications including power grid systems, epidemics, and sensor network.



Christos Faloutsos is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), the Research Contributions Award in ICDM 2006, the SIGKDD Innovations Award (2010), 25 “best paper” awards (including 5 “test of time” awards), and four teaching awards. Eight of his advisees or co-advisees have attracted KDD or SCS dissertation awards, He is an ACM Fellow, he has served as a member of the executive committee of SIGKDD; he has published over 350 refereed articles, 17 book chapters and two monographs. He holds seven patents (and 2 pending), and he has given over 40 tutorials and over 20 invited distinguished lectures. His research interests include large-scale data mining with emphasis on graphs and time sequences; anomaly detection, tensors, and fractals.

Correspondence and offprint requests to: Miguel Araujo, Senior Data Scientist at Feedzai, Portugal Email: miguelaraujo.cs@gmail.com