# TensorCast: Forecasting with Context using Coupled Tensors

Miguel Araujo
School of Computer Science
CMU and INESC-TEC
miguelaraujo.cs@gmail.com

Pedro Ribeiro
Computer Science Department
University of Porto and INESC-TEC
pribeiro@dcc.fc.up.pt

Christos Faloutsos
School of Computer Science
Carnegie Mellon University
christos@cs.cmu.edu

***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 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.**

## I. 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.

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:

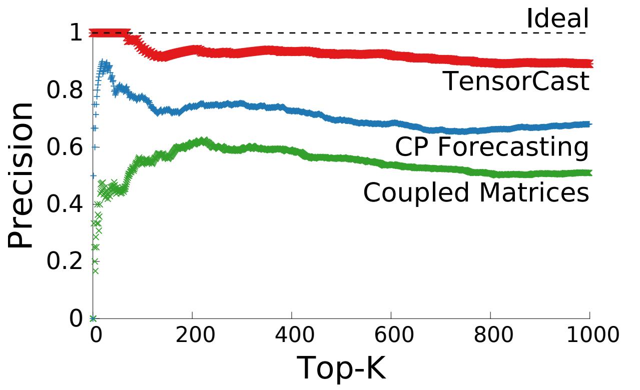
1) **Effectiveness**: TensorCast achieves over 20% higher precision in top-1000 queries and double the precision when finding new relations than comparable alternatives.
2) **Scalability** : TENSORCAST scales well ($E + N \log N$) with the input size and is tested in datasets with over *300M* interactions.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

## II. BACKGROUND

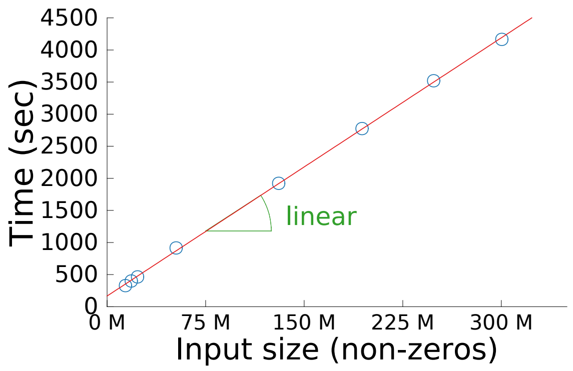
**Notation.** As common in the literature, we denote vectors by boldface lowercase letters (e.g., $\boldsymbol{a}$), matrices by boldface uppercase letters (e.g., $\boldsymbol{A}$) and tensors by boldface caligraphic letters (e.g., $\boldsymbol{\mathcal{X}}$). For convenience, we refer to the $f$-th 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 I for operators and additional symbols we use throughout the paper.

### A. Tensor Factorizations

Tensors are multidimensional arrays that generalize the concept of matrices. As a consequence, they are a popular choice when representing time-evolving relations, such as Facebook interactions [1], sensor networks [2] or EEG data for detecting the origin of epilepsy seizures [3]. When properly applied, tensor factorizations identify the underlying low-dimensional latent structure of the data. The latent factors are



(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**.

TABLE I
SYMBOLS AND DEFINITIONS

| Symbols | Definitions |
|---|---|
| $\|\mathcal{X}\|_F$ | Frobenius norm of tensor $\mathcal{X}$ |
| $\mathcal{X}_{(k)}$ | Mode-k matricization |
| $M^t$ | Matrix transpose |
| $\circ$ | Vector outer product |
| $\odot$ | Khatri-rao product |
| $\otimes$ | Hadamard (entrywise) product |
| $\oslash$ | Hadamard (entrywise) division |

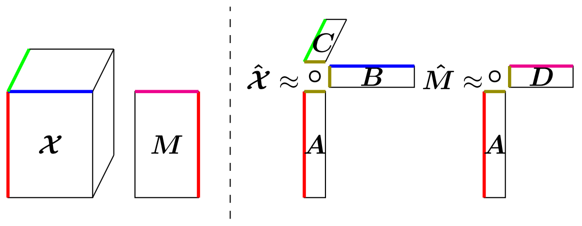


Fig. 2. A simple Coupled Matrix-Tensor Factorization.

then used to identify anomalies, to estimate missing values or to understand how the data was generated in the first place. The PARAFAC [4] (also called CP) decomposition is one of the most popular among the many tensor factorizations flavors [5], 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 $\mathcal{X}$ and the reconstructed tensor: $\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \left\| \mathcal{X} - \sum_f \boldsymbol{a_f} \circ \boldsymbol{b_f} \circ \boldsymbol{c_f} \right\|_F^2$.

### B. 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 $\mathcal{X}$ coupled with a matrix $\boldsymbol{M}$ on its first mode can be obtained by minimizing

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

where $\alpha$ 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:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{T}} \left\| \mathcal{X} - \hat{\mathcal{X}} \right\|_F^2 + \alpha \left\| \mathcal{Y} - \hat{\mathcal{Y}} \right\|_F^2 \quad (2)$$

where:

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

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

Many techniques have been proposed to solve this non-negative optimization problem, such as projected Stochastic Gradient Descent (SGD) [6] (i.e., additive update rules) and multiplicative update rules. Most of this work extends Lee and Seung's multiplicative matrix updates formulae [7] for matrices, notably the simple extension for tensors [8] and the many coupled extensions, e.g. Generalized Tensor Factorization [9], [10]. Update equations can be found in Appendix A.

### C. 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 [11], that eigenvectors of Kronecker graphs exhibit a multinomial distribution [12, theorem 3] and multiple generative models where power-law communities arise have been proposed [13], [14], [15]. 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` $\times$ `venue` $\times$ `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.

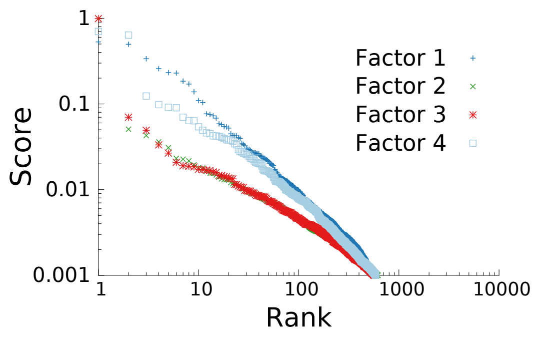


Fig. 3. **Scores of the factor vectors are highly skewed.** Non-negative factorization of the DBLP `author` $\times$ `venue` $\times$ `year` tensor. Note the logarithmic scale in both axis.

## III. RELATED WORK

### A. *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 [16] proposed the use of Cone Trees to speed-up this search. Other approaches map this problem into smaller sets of cosine-similarity searches [17], 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) [18], [19]. However, this is a non-convex optimization problem in general.

### B. *Link Prediction*

A large body of literature on link prediction has been created since its introduction [20]. 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 [21], [22], but there were also several approaches which showed that using the age of the link could lead to improved predictions [23].

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 [24], [25]) from methods that deal directly with the increased dimensionality, such as tensor-based methods. CP Forecasting [26] finds a low-rank PARAFAC factorization and forecasts the time-component in order to incorporate seasonality. TriMine [27] 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 [25] 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 II 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.

## IV. 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 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

TABLE II
TENSORCAST INTEGRATES CONTEXT AND TIME-AWARENESS.

| Property | Truncated SVD | Truncated Katz | Coupled Matrices (e.g., [25]) | VAR[28], ARIMA[29], etc. | CP Forecasting [26] | TriMine [27] | TENSORCAST |
|---|---|---|---|---|---|---|---|
| Scalability | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ |
| Interpretability | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ |
| Coupled setting |  |  | ✓ |  |  |  | ✓ |
| Time-awareness |  |  |  | ✓ | ✓ | ✓ | ✓ |
| Context-awareness |  |  | ✓ |  |  |  | ✓ |
| Forecasting |  |  |  | ✓ | ✓ | ✓ | ✓ |
| Ordered Forecasting |  |  |  | ✓ | ✓ |  | ✓ |

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:

**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.

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

### A. 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 reduces 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 V.

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

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

See Appendix A for further details.

### B. 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

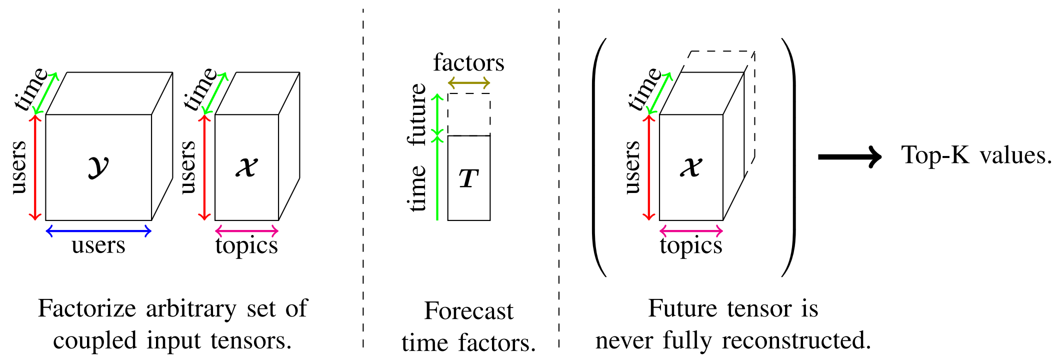


Fig. 4. **Overview** of TENSORCAST.

small set of $F$ dense factor vectors, hence easy to forecast, that will provide an approximation $\hat{\mathcal{X}}$ 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.

## *C. 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 $\hat{\mathcal{X}}$ 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.

[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 $\boldsymbol{a}_1\boldsymbol{b}_1\boldsymbol{s}_2$.

**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
1  while $\sum_i \boldsymbol{F}_i.next.factorScore \leq \boldsymbol{R}.last.Score$ do
2      f $\leftarrow \arg\max_i(\boldsymbol{F}_i.next.factorScore)$
3      element $\leftarrow \boldsymbol{F}_f.next$
4      $\boldsymbol{F}_f.pop$
5      $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup element.fullScore$
6      if $\boldsymbol{R}.size > K$ then
7          $\boldsymbol{R} \leftarrow \boldsymbol{R} - \arg\min(\boldsymbol{R})$
8      end
9  end
10 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. $\square$

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* $\mathbf{a}$ *and* $\mathbf{b}$ *follow power-laws with exponents* $\alpha_a$ *and* $\alpha_b$*, 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 [30, 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 II-C 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

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

[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.

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 $K$ 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.

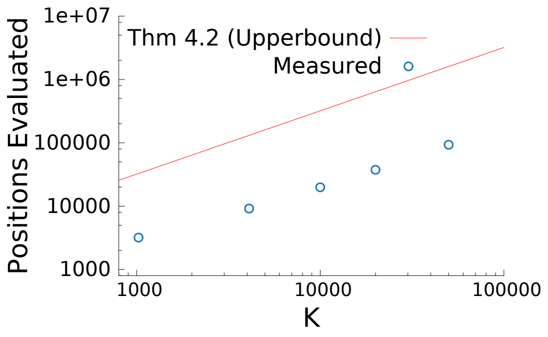


Fig. 5. TENSORCAST **only checks a linear number of elements** of the tensor.

## D. *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.

## V. 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 III
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 decrease as we forecast farther to the future?

TENSORCAST is tested on two big datasets detailed in Table III. 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 interest (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 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.

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

[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.

The competing *CP Forecasting* [26] 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.

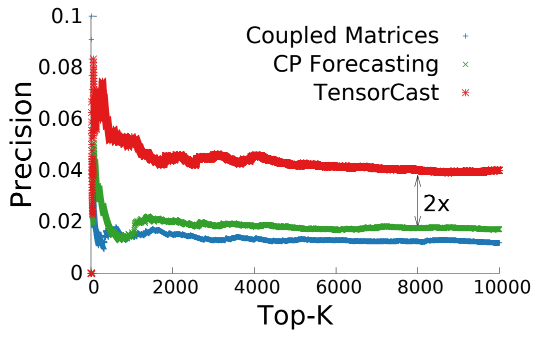


Fig. 6. **Double precision** when forecasting **novel** $(user, hashtag)$ relations in the TWITTER tensor.

### *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.

### *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

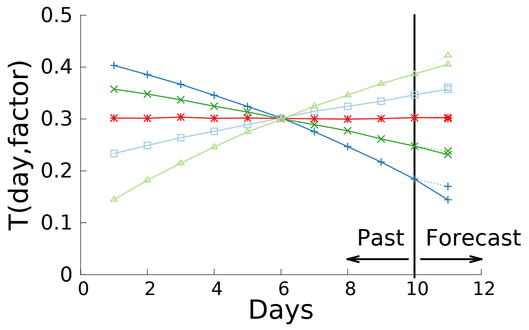


Fig. 7. TENSORCAST correctly **forecasts growth and decay** of groups in synthetic data. [dashed - forecast; solid - real]

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.

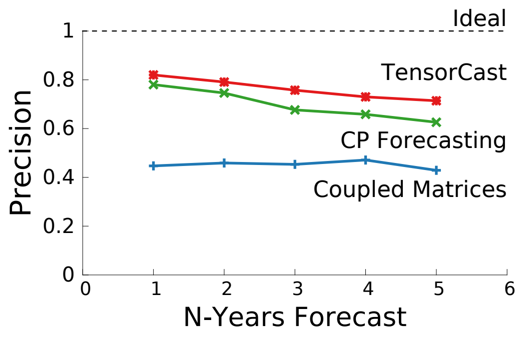


Fig. 8. TENSORCAST achieves higher precision at every forecasting horizon.

*Discoveries - TensorCast at work*

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 Figure 10 (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 9 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.

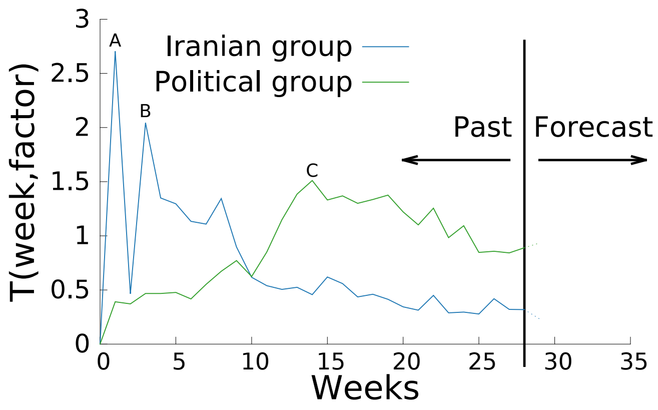


Fig. 9. Increased precision is achieved by grouping interactions. [A-Start of 2009 election protests; B-President Obama references Neda Soltan, killed in the protests; C-Taxpayer March in Washington]

**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 [31].
2) Forecasts were done using Gnu's Regression, Econometrics and Time-series Library (GNU gretl) [32].
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/.

## VI. CONCLUSIONS

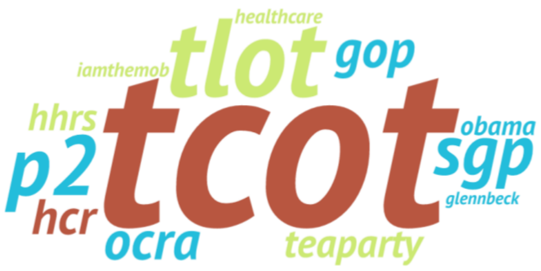
We presented TensorCast, a method which addresses the forecasting problem on big time-evolving datasets when contextual information is available. We leverage typical graph properties in order to create a linearithmic algorithm that can find novel relations in very big datasets efficiently.

The main advantages of our method are:

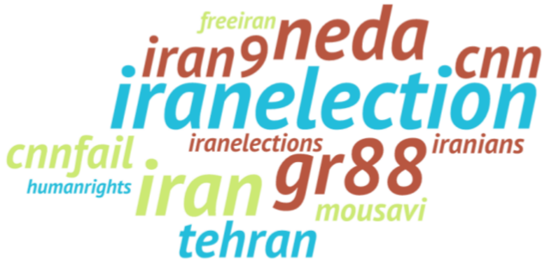
1) **Effectiveness**: TensorCast achieves over 20% higher precision in top-1000 queries and double the precision when finding new releations than comparable alternatives.
2) **Scalability** : TENSORCAST scales linearithmically with the input size and is tested in datasets with over *three hundred million* 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.

## ACKNOWLEDGMENT

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



(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. 10. TENSORCAST **finds and forecasts groups with similar interests** on TWITTER.

the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by FCT  Fundao para a Cilncia 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.

## REFERENCES

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

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

[3] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener, "Multiway analysis of epilepsy tensors," *Bioinformatics*, vol. 23, no. 13, pp. i10–i18, 2007.

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

[5] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

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

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

[8] M. Welling and M. Weber, "Positive tensor factorization," *Pattern Recognition Letters*, vol. 22, no. 12, pp. 1255–1261, 2001.

[9] Y. K. Yılmaz, "Generalized tensor factorization," Ph.D. dissertation, Citeseer, 2012.

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

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

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

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

[14] X. Zhou, L. Xiang, and W. Xiao-Fan, "Weighted evolving networks with self-organized communities," *Communications in Theoretical Physics*, vol. 50, no. 1, p. 261, 2008.

[15] Z. Xie, X. Li, and X. Wang, "A new community-based evolving network model," *Physica A: Statistical Mechanics and its Applications*, vol. 384, no. 2, pp. 725–732, 2007.

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

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

[18] A. Shrivastava and P. Li, "Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips)," *arXiv preprint arXiv:1410.5410*, 2014.

[19] B. Neyshabur and N. Srebro, "On symmetric and asymmetric lshs for inner product search," *arXiv preprint arXiv:1410.5518*, 2014.

[20] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

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

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

[23] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

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

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

[26] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 2, p. 10, 2011.

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

[28] A. Zellner, "An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias," *Journal of the American statistical Association*, vol. 57, no. 298, pp. 348–368, 1962.

[29] G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal*

*of the American statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[30] G. Grimmett and D. Stirzaker, *Probability and random processes*. Oxford university press, 2001.

[31] B. W. Bader, T. G. Kolda *et al.*, "Matlab tensor toolbox version 2.6," Available online, February 2015. [Online]. Available: http://www.sandia.gov/~tgkolda/TensorToolbox/

[32] G. Baiocchi and W. Distaso, "Gretl: Econometric software for the gnu generation," *Journal of applied econometrics*, vol. 18, no. 1, pp. 105–110, 2003.

[33] Z. He, S. Xie, R. Zdunek, G. Zhou, and A. Cichocki, "Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2117–2131, 2011.

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

## Appendix
## Multiplicative Updates of Coupled Tensors Factorization

The non-negative coupled tensor factorization problem

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

is well studied and its multiplicative update equations have been previously described in the literature (e.g., considering the dispersion parameter $\alpha$ [10]). The solution can be found by iteratively updating

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

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

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

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

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 [8] note the need for a scaling exponent (for the simple, non-coupled case):

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

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 [33]. 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 [34]).

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