TensorCast: Forecasting Time-Evolving Networks with Contextual Information

Miguel Araújo¹,², Pedro Ribeiro¹ and Christos Faloutsos²
¹ University of Porto and CRACS/INESC-TEC
² School of Computer Science, Carnegie Mellon University
miguelaraujo.cs@gmail.com, pribeiro@dcc.fc.up.pt, christos@cs.cmu.edu

Abstract
Can we forecast future connections in a social network? Can we predict who will start using a given hashtag in Twitter, leveraging contextual information such as who follows or retweets whom to improve our predictions? In this paper we present an abridged report of TensorCast, a method for forecasting time-evolving networks, that uses coupled tensors to incorporate multiple information sources. TensorCast is scalable (linearithmic on the number of connections), effective (more precise than competing methods) and general (applicable to any data source representable by a tensor). We also showcase our method when applied to forecast two large scale heterogeneous real world temporal networks, namely Twitter and DBLP.

1 Introduction
If a group has been discussing the #elections on Twitter, can we predict who is going to join the discussion next week? Intuitively, we should take into account not only other used hashtags (#), but also user to user interactions such as followers and retweets. Similarly, can we forecast who is going to publish on a given conference next year? We should consider not only previous conferences where the author published, but also user to user information such as co-authorships.

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. We propose to model these heterogeneous evolving 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 leveraged 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.

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 we avoid this? How can we obtain the top most likely interactions without iterating through them all?

We propose TensorCast, a tensor forecasting method that is able to merge multiple data sources in order to predict future membership relations. Our method combines a non-negative coupled factorization of generalized tensors with an efficient algorithm that is able to obtain the top-K records from an existing factorization, without reconstructing the full tensor. We achieve over 20% higher precision in top-1000 queries and double the precision when finding new relations than comparable alternatives and our method scales well (linearithmically) with the input size (we show experimental results in graphs with over 300M connections).

TensorCast is open-source and can be obtained at http://www.dcc.fc.up.pt/~pribeiro/tensorcast/ This is abridged version of a longer paper which got the ICDM’2017 best paper award [Araújo et al., 2017].

2 Background and Related Work

Notation. As commonly used, we denote vectors by boldface lowercase letters (e.g., a), matrices by boldface uppercase letters (e.g., A) and tensors by boldface calligraphic letters (e.g., Χ). We refer to the f-th column of A as af and to the (i, j, k) entry of 3-mode tensor Χ as Χijk. M denotes a matrix transpose, ||Χ||F denotes the Frobenius norm of tensor Χ, and Χ its mode-k matricization. The vector outer product is denoted by ⊙ and the Khatri-rao product by ∘.

2.1 Tensor Factorizations

Tensors are multidimensional arrays that generalize the concept of matrices, and are a popular choice when representing time-evolving relations, such as Facebook interactions [Papalexakis et al., 2012] or sensor networks [Sun et al., 2006]. When properly applied, tensor factorizations identify the underlying low-dimensional latent structure of the data, which can be used for tasks such as identifying anomalies or estimate missing values. The PARAFAC [Harshman, 1970] (also called CP) decomposition is one of the most popular among the many tensor factorizations flavors [Kolda et al., 2017], as it factorizes a tensor into a sum of rank-1 tensors. In three modes, the problem is usually framed as finding factor matrices A, B and C that minimize the squared error between Χ and the reconstructed tensor:

\[ \min_{A,B,C} \| \sum_{f} a_f \odot b_f \odot c_f \|_F^2 \].
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 $\mathcal{X}$ coupled with a matrix $M$ on its first mode can be obtained by minimizing

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

where $\alpha$ is a parameter representing the strength of the coupling, i.e., how important $M$ is to improve the prediction.

![Figure 1: A simple Coupled Matrix-Tensor Factorization.](image)

The matrix part of the Coupled Matrix-Tensor Factorization depicted in Fig. 1 is useful to model additional static information about one of the modes of the tensor of interest. Whenever the side information available is time-evolving, a model where two tensors are coupled along one of the dimensions is more appropriate, preserving the time component:

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

where $\hat{\mathcal{X}} = \sum_f a_f \circ b_f \circ t_f$ and $\hat{Y} = \sum_f a_f \circ c_f \circ t_f$

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 work extends Lee and Seung’s multiplicative matrix updates formula [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 [Yilmaz, 2012; Simsekli et al., 2013].

2.3 Skewed Building Blocks

When factorizing real-life graphs, non-negative factors scores 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 [Araujo et al., 2014] and that eigenvectors of Kronecker graphs exhibit a multinomial distribution [Leskovec et al., 2005]. TENSORSFCAST leverages this property in order to speed-up its computation of Top-$K$ elements without reconstructing the forecasted tensor.

2.4 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 which can be stated as: given matrices $A$ and $B$ of sizes $N \times F$ and $M \times F$, find the top-$K$ $(i,j)$ pairs of the $AB^T$ matrix product. A 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, but this is a non-convex optimization problem in general.

2.5 Link Prediction

In the classical structural link prediction problem [Liben-Nowell and Kleinberg, 2007] the goal is to predict which links are more likely to appear, assuming that links are never or seldom removed. With time-evolving graphs, temporal link prediction methods have been developed to predict future snapshots, where links are not guaranteed to persist over time. In this setting we distinguish methods that rely on collapsing (matricizing) the input data (e.g., exponential decay of edge weights [Sharan and Neville, 2008]) 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. TriMine [Matsubara et al., 2012] similarly factorizes the input tensor, but then applies probabilistic inference in order to identify hidden topics. These methods are not able to integrate contextual information. Other approaches integrate structure and content in the same prediction task, e.g. Gao et al [Gao et al., 2011] suggest a coupled matrix factorization and graph regularization technique to obtain the latent factors after an exponential decay of the temporal network. Another approach is to use time series analysis and regression based methods such as VAR [Zellner, 1962] or ARIMA [Box and Pierce, 1970].

However, none of these methods fulfills all the requirements for forecasting when contextual information is considered. Table 1 contrasts TENSORSFCAST 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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Truncated SVD</th>
<th>Truncated Katz</th>
<th>Coupled Matrices</th>
<th>VAR, ARIMA</th>
<th>CP Forecasting</th>
<th>TriMine</th>
<th>TENSORSFCAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
</tr>
<tr>
<td>Interpretability</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
</tr>
<tr>
<td>Coupled setting</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
</tr>
<tr>
<td>Time-awareness</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
</tr>
<tr>
<td>Context-awareness</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
</tr>
<tr>
<td>Forecasting</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
</tr>
<tr>
<td>Ordered Forecasting</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
<td>ü</td>
</tr>
</tbody>
</table>

Table 1: TENSORSFCAST integrates context and time-awareness.
3 Proposed Approach: TENSORCAST

We assume a coupled-tensors setting where multiple tensors are related by a common mode. One of these tensors should be a 3-dimensional binary tensor, with one mode corresponding to a time component which we would like to forecast. There are many scenarios that fall under this setting. For example, we could have a tensor of membership records (user, topic, time) and an additional collection of user interaction records (user, user, time), with a possible forecasting problem of predicting which users will interact with which topics, taking advantage of the both sources of information.

We tackle the following general tensor forecasting problem: given coupled tensors (X and Y), a number of K relations and S time-steps, forecast the ranked list of K most likely non-zero elements of X in the next S time-steps.

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 2 illustrates the intuition of our method.

3.1 Non-negative Coupled Factorization

Consider that the tensor of interest, 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 X to be big and sparse. Factorizing the input data reduces the number of elements to be forecasted, but also, and 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.

We explore how user interactions can be leveraged to improve forecasting of user-entity relations. We model the problem as two coupled tensors where tensor 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:

\[
A \leftarrow A \odot \frac{X(1)B \odot T + \alpha Y(1)A \odot T}{A(B \odot T)\text{tr}(B \odot T) + \alpha A(A \odot T)\text{tr}(A \odot T)}
\]

3.2 Forecasting

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

3.3 Tensor Top-K elements

The forecast of the next time-step is a \( N \times M \times S \) tensor represented as \( \sum a_f \otimes b_f \otimes s_f \) where A is \( N \times F \), B is \( M \times F \) and 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 performed on non-random graph that follows typical properties. The goal is to identify the \( K(i,j,k) \) positions with highest value \( \sum A_{ij}B_{jj}s_{kj} \).

Top-K of a single factor. We start by creating a data structure to get the next biggest element in \( O(\log(SM)) \) time, with \( O(S\log S + M\log M + N\log N + SM) \) preprocessing. Firstly, we sort the vectors (s, a and b) in decreasing order. We now know that the biggest element is given by \( a_1b_1s_1 \), and an element \( a_ibJs_k \) only needs to be considered after \( a_{i-1}b_{j-1}s_k \) and \( a_ibjs_{k-1} \) have all been identified as one of the biggest \( K \). Hence, we can create a priority queue which 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 is efficient under realistic scenarios. We keep a list \( R \) of the \( K \) biggest positions evaluated so far. 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 \( R \).

With this, we check at most \( KS^1+\pi \) elements if every frontal slice \( aF \odot bF \) follows a power-law (see [Araujo et al., 2017] for a detailed description of the algorithm and respective proofs). This means TENSORCAST is linear on the number of elements we want to obtain times the number of time-steps forecasted and 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. Empirical evaluation on real datasets confirmed our bounds.
4 Experiments

**TensorCast** is tested on two big datasets detailed in Table 2. In DBLP we have a tensor 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 data. In Twitter, the main tensor relates users and hashtags they used from June to December 2009, while retweets are used as the auxiliary tensor. Tweets are grouped weekly and evaluation is performed on week 51.

Every factorization uses 10 factors (both ours and competing methods). In Twitter, we weighted the reconstruction of the main tensor 20x more relevant than the context tensor, and in DBLP 2.66x higher (so that both tensors have the same reconstruction error when considering empty factors).

**How fast is TensorCast?** We start by evaluating our method’s scalability in Twitter when changing the number of non-zeros in both tensors. 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 3 shows TensorCast’s linear scalability. Similar results were obtained for DBLP.

**How effective is TensorCast?** Figure 4 showcases TensorCast’s accuracy on the task of predicting top-K relations on future time steps, as we increase K on the DBLP dataset (similar results were obtained for Twitter).

Note the importance of TensorCast’s ability of being simultaneously contextual and time-aware (competing state-of-the-art is limited due to ignoring either one of these aspects). The competing CP Forecasting was run using Holt forecasting, given the lack of data seasonality. The results of Coupled Matrices were obtained by finding non-negative factors that minimize the reconstruction error of the collapsed tensors, weighted for the same importance.

Considering only the harder problem of predicting new relationships (that didn’t exist on any previous time step), the overall precision decreases, but the advantage over the competing methods is even higher (we double their precision).

### Table 2: Summary of real-world networks used.

<table>
<thead>
<tr>
<th>Users</th>
<th>Groups</th>
<th>Timesteps</th>
<th>Memberships</th>
<th>Interactions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1734902</td>
<td>5476</td>
<td>79</td>
<td>8049559</td>
<td>21423244</td>
<td>DBLP - venues published and co-authorships.</td>
</tr>
<tr>
<td>12426133</td>
<td>2326843</td>
<td>31</td>
<td>30281817</td>
<td>282280158</td>
<td>Twitter - hashtags used and retweets.</td>
</tr>
</tbody>
</table>

**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 each method’s precision. Figure 5 shows the average precision for each forecasting horizon.

In addition to forecasting, the groups found by TensorCast are interpretable due to the non-negativeness of the factors (ex: in Twitter one of the groups we found was characterized by conservative political hashtags, and another one was related to human rights and the iranian elections).

### 5 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 precise algorithm that can find novel relations in very big datasets efficiently.

![Figure 3: TensorCast scales linearly with the number of non-zeros in the Twitter dataset.](image)

![Figure 4: Higher precision vs competing methods when forecasting (author, venue) relations in the DBLP dataset.](image)

![Figure 5: TensorCast achieves higher precision at every forecasting horizon in the DBLP dataset.](image)
Acknowledgements

This work was partly financed by the ERDF through the COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EIA/50014/2013.

This material is based upon work supported by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053.

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 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


