

# Link Prediction on Evolving Data using Tensor Factorization

Stephan Spiegel<sup>1</sup>, Jan Clausen<sup>1</sup>, Sahin Albayrak<sup>1</sup>, and Jérôme Kunegis<sup>2</sup>

<sup>1</sup> DAI-Labor, Technical University Berlin  
Ernst-Reuter-Platz 7, 10587 Berlin, Germany  
{stephan.spiegel, jan.clausen, sahin.albayrak}@dai-labor.de  
<http://www.dai-labor.de>

<sup>2</sup> University of Koblenz-Landau  
Universitätsstraße 1, 56072 Koblenz, Germany  
kunegis@uni-koblenz.de  
<http://west.uni-koblenz.de/>

**Abstract.** Within the last few years a lot of research has been done on large social and information networks. One of the principal challenges concerning complex networks is link prediction. Most link prediction algorithms are based on the underlying network structure in terms of traditional graph theory. In order to design efficient algorithms for large scale networks, researchers increasingly adapt methods from advanced matrix and tensor computations.

This paper proposes a novel approach of link prediction for complex networks by means of multi-way tensors. In addition to structural data we furthermore consider temporal evolution of a network. Our approach applies the canonical *Parafac* decomposition to reduce tensor dimensionality and to retrieve latent trends.

For the development and evaluation of our proposed link prediction algorithm we employed various popular datasets of online social networks like *Facebook* and *Wikipedia*. Our results show significant improvements for evolutionary networks in terms of prediction accuracy measured through mean average precision.

**Keywords:** Link Prediction Algorithm, Temporal Network Analysis, Evolving Data, Multi-way Array, Tensor Factorization

## 1 Introduction

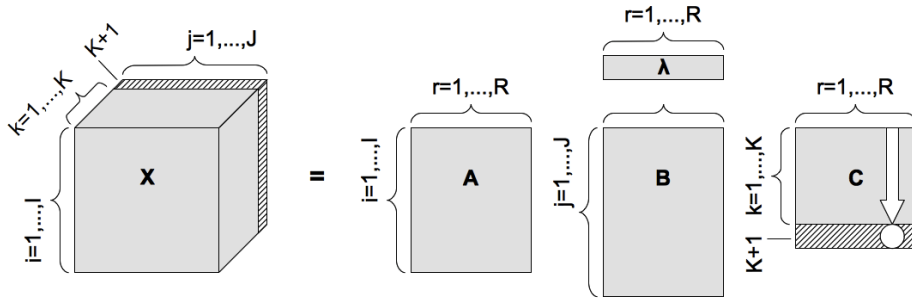
The growing interest in large-scale networks is originated from the increasing number of online social platforms and information networks. Many studies have scrutinized static graph properties of single snapshots, which lacks of information about network evolution [11, 14]. Especially for the challenge of link prediction it is necessary to observe trends within time, which are determined by the addition and deletion of nodes. Most network evolution models are based on the

average node degree or the growing function of the network size [12–16, 18]. In our approach we capture temporal trends within a multi-way array, also known as tensor, where time is represented as a separate dimension.

Multi-way data analysis is the extension of two-way data analysis to higher-order data, and is mostly used to extract hidden structures and capture underlying correlations between variables in a multi-way array [3]. Instead of only considering pairwise relations between variables (e.g. {user, item}, {user,time} and {item,time}), multi-way data analysis attempts to discover latent factors among multiple objects [5, 22]. The most popular multi-way models in literature are *Tucker* and *Parafac*, which both factorize higher-order data into low-dimensional components [3].

This study employs the *Parafac* model to estimate the probability of an edge to be added within an observed network. The examined data is mainly derived from online social and information networks, and is commonly represented as  $i \times j \times k$  multi-way array; where  $i$  and  $j$  represent the number of users or items respectively, and  $k$  denotes the number of points in time capturing the date of interaction. Our main purpose is to predict whether and how much a user is likely to rate an item and to which probability a user is going to interact with someone else. However there are many other applications for link prediction, such as predicting the web pages a web surfer may visit on a given day, the places that a traveler may fly to in a given month, or the patterns of computer network traffic [1, 4, 7, 21].

The rest of the paper is organized as follows. Section 2 gives a general introduction to tensor factorization, whereas Section 3 describes our own link prediction algorithm. The experimental results are presented in Section 4. Finally, Section 5 concludes this paper.



**Fig. 1.** CP Decomposition of a 3rd-order Tensor:  
 $X \in \mathbb{R}^{I \times J \times (K+1)} \Rightarrow \lambda \in \mathbb{R}^R; A \in \mathbb{R}^{I \times R}; B \in \mathbb{R}^{J \times R}; C \in \mathbb{R}^{(K+1) \times R}$

## 2 Tensor Decomposition

An  $N$ th-order tensor is the product of  $N$  vector spaces, each of which has its own coordinate system [9]. Usually first-order and second-order tensors are referred to as vectors and matrices respectively. This paper mainly investigates third-order tensors, also known as multi-way arrays, which can be imagined as a three-dimensional cube like illustrated in Figure 1.

Tensor decompositions first emerged in psychometrics and chemometrics, but recently were adopted to other domains, including data mining and graph analysis. They aim to reveal underlying linear structures and are often employed to reduce noise or data dimensionality [3, 9]. Factorization techniques like the CP (CanDecomp/ParaFac) and Tucker model can be considered higher-order generalization of the matrix singular value decomposition (SVD) and principal component analysis (PCA) [1, 9]. However, the CP model is more advantageous in terms of interpretability, uniqueness of solution and determination of parameters [3]. Following Kolda [9] the CP mode-3-model can be expressed as either a sum of rank-one tensors (each of them an outer product of vectors  $a_r$ ,  $b_r$ ,  $c_r$  and a weight  $\lambda_r$ ) or factor matrices (refer to Figure 1):

$$X \approx \sum_{r=1}^R \lambda_r (a_r \circ b_r \circ c_r) \equiv [\lambda; A, B, C] \quad (1)$$

with  $X$  representing our original tensor and  $R$  specifying the number of rank-one components. For the derivation of our proposed link prediction model we use the equivalent factor representation (1), where all columns of matrices  $A$ ,  $B$ , and  $C$  are normalized to length one with the weights absorbed into the vector  $\lambda \in \mathbb{R}^R$ . Each of the factor matrices refer to a combination of the vectors from the rank-one components, i.e.

$$A = [a_1 \ a_2 \ \cdots \ a_R] \equiv \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,R} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I,1} & a_{I,2} & \cdots & a_{I,R} \end{bmatrix} \quad (2)$$

and likewise for  $B$  and  $C$ . In terms of frontal slices the introduced three-way model (1) can also be written as [9]:

$$X_k \approx AD^{(k)}B^T \quad \text{with} \quad D^{(k)} = \text{diag}(\lambda, c_{k,:}) \quad (3)$$

If we consider a third-order tensor which captures intercommunication patterns of  $i$  users over  $k$  periods of time ( $X \in \mathbb{R}^{I \times I \times K}$ ), the  $k$ -th frontal slide would comprise all information about user interactions of date  $k$  [2]. Our own link prediction approach estimates future user correlations ( $K + 1$ ) by analysing the temporal trend kept in the observed time slices.

Note that, according to the *Spectral Evolution Model* [10], Equation (3) can also be considered as a joint decomposition of the respective time slice matrices. In the course of time the spectrum captured in matrix  $D$  will change, whereas the eigenvectors of matrices  $A$  and  $B$  stay the same [10].

### 3 Link Prediction

As might be expected, the term link prediction is characterized as prognosis of future edges within complex networks, where edges may express relations between individuals or abstract objects respectively [16, 20]. In general, link prediction techniques are grouped into neighbor-based and link-based algorithms, which either consider adjacent nodes (e.g. *Common Neighbors & Preferential Attachment*) or paths between nodes (e.g. *Katz Algorithm*) to estimate the probability of future edges [16]. Recently there emerged some novel approaches that make use of psychological concepts (e.g. *Theory of Balance & Theory of Status*) to predict links [12, 13]. Another group of semantic algorithms utilize attributes of nodes to make prognostications [15, 18]. However, many popular link prediction techniques merely scrutinize static graph properties of single snapshots, giving a lack of information about network evolution [11, 14].

Our own link prediction approach differs from common techniques in that we consider multiple network snapshots with temporal trends captured in the time factor matrix  $C$ , given by the previously introduced CP tensor decomposition (refer to Equation 1 and Figure 1). Furthermore, we employ exponential smoothing to extrapolate future points in time based on the latent temporal trends retrieved by the tensor factorization. The combination of canonical *Parafac* decomposition with exponential smoothing was inspired by *E. Acar* and *T. G. Kolda* [1], who suggest the *Hold-Winters* method to analyse trends and seasonality of temporal data captured in tensors.

In statistics, exponential smoothing is a technique that can be applied to time series data, either to produce smoothed data for presentation, or to make forecasts. The time series data themselves are a sequence of observations. The raw data sequence is often represented by  $x_t$ , and the output of the exponential smoothing algorithm is commonly written as  $s_t$  which may be regarded as our best estimate of what the next value of  $x$  will be. When the sequence of observations begins at time  $t = 0$ , the simplest form of exponential smoothing is given by the expressions:

$$\begin{aligned} s_1 &= x_0 \\ s_{t+1} &= \alpha x_t + (1 - \alpha)s_t \end{aligned} \quad (4)$$

where  $\alpha$  is the smoothing factor ( $0 < \alpha < 1$ ), which assigns exponentially decreasing weights over time. For further examinations we balance the influence of most recent terms in time series and older data on the extrapolated points in time ( $\alpha = 0.5$ ). By direct substitution of the defining Equation (4) for simple exponential smoothing back into itself we find that:

$$\begin{aligned} s_{t+1} &= \alpha x_t + (1 - \alpha)s_t \\ &= \alpha x_t + \alpha(1 - \alpha)x_{t-1} + (1 - \alpha)^2 s_{t-2} \\ &= \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \alpha(1 - \alpha)^2 x_{t-2} + \dots + (1 - \alpha)^t x_0 \end{aligned} \quad (5)$$

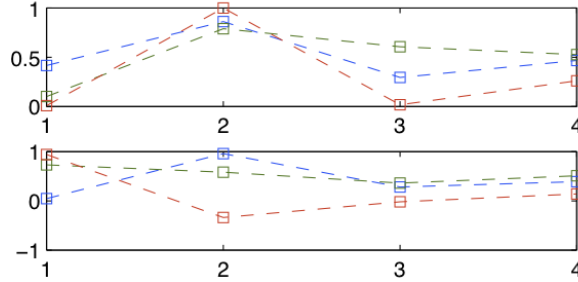
In our link prediction algorithm we employ the exponential smoothing technique to estimate the  $K + 1$ -th frontal slice of tensor  $X$ , containing information about future correlations between the variables held in the first and second mode (e.g. users or items). Inferred from Equation (3), computation of the  $K + 1$ -th frontal slice requires to predict vector  $c_{K+1,:}$ , which can be derived from the temporal trend captured in the columns of factor matrix  $C$  (as shown in Figure 1 & 2). The exponential smoothing for each entry of vector  $c_{K+1,:}$  can be formulated as:

$$\begin{aligned} c_{K+1,r} &= \alpha c_{K,r} + \alpha(1 - \alpha)c_{K-1,r} + \alpha(1 - \alpha)^2 c_{K-2,r} \\ &\quad + \alpha(1 - \alpha)^3 c_{K-3,r} + \dots + (1 - \alpha)^K c_{1,r} \quad (6) \\ &(r = 1, \dots, R \text{ and } 0 < \alpha < 1) \end{aligned}$$

Given the extrapolated vector  $c_{K+1}$ , we are not only able to estimate the  $K + 1$ -th frontal slice (refer to Equation 3), but also single links  $x_{i,j,K+1}$  that are likely to be established in the future:

$$x_{i,j,K+1} \approx \sum_{r=1}^R \lambda_r (a_{i,r} \cdot b_{j,r} \cdot c_{K+1,r}) \quad (7)$$

However, we need to keep in mind that the CP model is an approximation of our original data [2]. The precision of the estimated links depends on various factors like data composition and sparsity, the range of values as well as the number of rank-one components, which are all discussed in the following section.



**Fig. 2.** Temporal Trends in Time Factor Matrix of Wiki-Edit-Ar and Facebook-Links dataset, where the 4th point is extrapolated by Exponential Smoothing

## 4 Evaluation

In order to evaluate the precision of our link prediction algorithm we need to split the observed data into a training set, which is used as input for the tensor analysis, and a test set, which is compared against the estimated results. The split is either based on a predefined percentage or can be done according a determined point in time. As a matter of course the overall precision depends on the proportion of training and test data. In our case three-quarter of the data are used for training and one-quarter for testing ( $\frac{75}{25}\%$ ).

Note that usually just a definite number of random entries ( $E$ ) contained in the test set are used for evaluation. Due to the fact that our link prediction algorithm also estimates missing edges, the random selection needs to incorporate the same amount of both zero and non-zero entries ( $E = E^- \cap E^+$ , with  $\|E^-\| = \|E^+\|$ ). Table 1 illustrates the split of a sample dataset, which comprises user-item tags and their corresponding time stamps.

Assuming our sample data of Table 1, the training set is used to construct a three-way array capturing tags on items given by users at a specific point in time ( $X \in \mathbb{R}^{I \times J \times K}$ ). Based on the previously introduced mathematical model we are now able to make predictions for tags situated in the future ( $[K + 1]$ -th frontal slice). In order to calculate the precision of our link prediction algorithm, we can compare the estimated values with those contained within our test set

(for both known and missing edges).

	UserID	ItemID	TimeStamp	Tag
Training Set	20	1747	200610	1
	21	5524	200712	1
	25	5067	200705	1
	31	1091	200611	1
75% Observed	⋮	⋮	⋮	⋮
Test Set $E^+$	20	7438	200803	1
	21	5525	200807	1
	⋮	⋮	⋮	⋮
$X$ of 25% Obs.	⋮	⋮	⋮	⋮
Test Set $E^-$	31	6388	200812	0
	43	5411	200804	0
$X$ of 25% Obs.	⋮	⋮	⋮	⋮

**Table 1.** Split of Sample Dataset

As might be expected, the accuracy of link prediction also vary according to the precision measure chosen. Due to its robustness *Mean Average Precision (MAP)* is a frequently used measure in the domain of information retrieval (IR) and machine learning (ML). Contrary to most single-value metrics which are based on an unordered list of values, *MAP* emphasizes ranking relevant values higher. The reordering of an observed user vector ( $X_{i:,K+1}$  for  $i = 1, \dots, I$ ) according the estimates is illustrated in Table 2.

Index	1	2	3	4	⇒	2	4	1	3
Observed	0	1	0	1		1	1	0	0
Estimated	.5	.9	.3	.8		.9	.8	.5	.3
	unordered					reordered			

**Table 2.** Reordered Sample User Vector

Given a reordered user vector we are able to calculate its average precision by summing up the precision values for each cut-off point in the sequence of ranked results. Average precision can be formulated as:

$$AP = \frac{1}{N} \sum_{r=1}^N P(r) \quad (8)$$

where  $r$  is the rank,  $N$  is the number of cut-off points (alias non-zero entries) and  $P(r)$  precision<sup>3</sup> at a given cut-off rank. The mean of all average precision values for each user vector equals our final measure called *MAP*.

All examined datasets and their according properties are listed in Table 3. Besides different information networks like *Wikipedia* and *Bibsonomy*, we furthermore scrutinized rating data of *MovieLens* and *Epinions*, social network data of *Facebook* as well as *Enron* communication data and other graph like data about *Internet-Growth*. In general, we can classify these datasets into bipartite graphs, which capture correlations between two different sets of elements (e.g. users & items), and unipartite graphs, which give information about interrelation of elements from a single set. Related to tensors, sets of elements are regarded as separate modes, or in other words dimensions. The categorie of the scrutinized datasets (bipartite or rather unipartite graph) can be infered from the structure column given in Table 3.

DATASET	STRUCTURE	#ENTRIES	#MODE1	#MODE2	#SLICES
wiki-edit-viwiki	$user \times page \times time$	2262679	13766	303867	$\sim 4$
wiki-edit-skwiki	$user \times page \times time$	2526392	7229	215638	$\sim 6$
wiki-edit-glwiki	$user \times page \times time$	1315066	2850	91594	$\sim 7$
wiki-edit-elwiki	$user \times page \times time$	1569075	8049	97149	$\sim 8$
wiki-edit-arwiki	$user \times page \times time$	4000735	25692	510033	$\sim 4$
movielens-10m-ut	$user \times tags \times time$	95580	2795	12553	$\sim 4$
movielens-10m-ui	$user \times item \times time$	95580	3097	6367	$\sim 8$
movielens-10m-ti	$tags \times item \times time$	95580	12775	6190	$\sim 4$
internet-growth	$page \times page \times time$	104824	20689	20689	$\sim 3$
facebook-wosn-wall	$user \times user \times time$	876993	30839	30839	$\sim 14$
facebook-wosn-links	$user \times user \times time$	1545686	57356	57356	$\sim 13$
epinions	$user \times user \times time$	19793847	91596	91596	$\sim 5$
enron	$user \times user \times time$	1149884	64145	64145	$\sim 9$
bibsonomy-2ut	$user \times tags \times time$	2555080	4804	167963	$\sim 8$
bibsonomy-2ui	$user \times item \times time$	2555080	1345	335335	$\sim 4$
bibsonomy-2ti	$tags \times item \times time$	2555080	155264	571768	$\sim 2$

**Table 3.** Properties of Examined Datasets

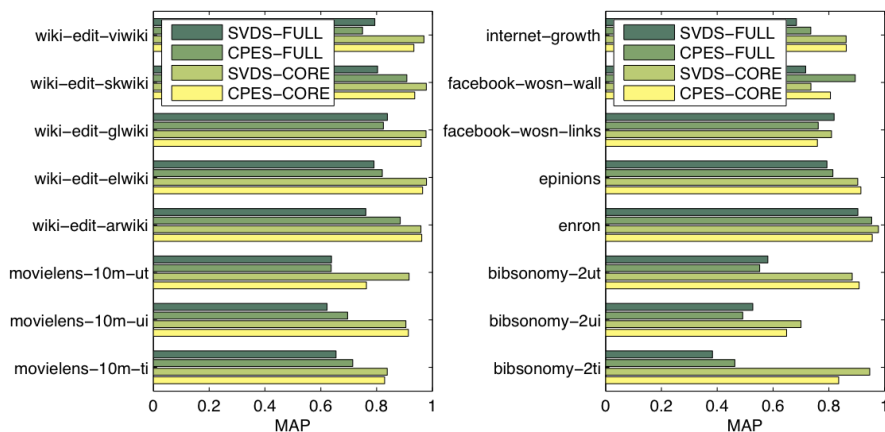
As mentioned earlier, this study mainly concentrates on three-dimensional datasets, with time being modeled in one of the modes. In order to make reliable link predictions for the future, we need to reduce the sparsity of our three-way array by squeezing the time dimension. Assuming that the precision relies upon connectedness of the examined graph, we limit the number of time slices in our tensor model by the following equation:

<sup>3</sup> Fraction of the elements retrieved that are relevant to the user’s information need.

$$\#slices = \frac{\#entries}{2 * \max(\#mode1, \#mode2)} \quad (9)$$

Typically a graph is connected if each node has a degree of at least two [8], implying that each fiber [9] of the first and second mode should have at minimum two non-zero entries (satisfied by Equation 9). Note that the number of slices correlates with the maximum number of possible rank-one components in our CP model (represented by variable  $R$  in Equation 1).

In addition to time compression (9) we furthermore ensure connectedness of a graph via reduction to its core component [11, 14]. By cutting off all periphery nodes with weak connections to the core, we are able to further improve the precision of our link prediction approach. This is due to the fact that we cannot infer something meaningful about a periphery node without an existing path to the core component of the observed network.



**Fig. 3.** Precision of Link Prediction

The mean average precision of our proposed link prediction algorithm (*CPES* alias *Canonical Parafac Decomposition with Exponential Smoothing*) for each examined dataset is illustrated in Figure 3. In order to compare the performance of our approach with an appropriate baseline algorithm, we furthermore scrutinized a matrix factorization method using *Singular Value Decomposition (SVD)*, which has been proven to be an effective technique in many data applications [1, 6, 17, 19]. For the evaluation of the matrix *SVD* technique it is necessary to collapse the time dimension of our original three-dimensional tensor, resulting in a flat two-dimensional structure (represented by  $M$ ) that accumulates all entries of the first and second mode.

*SVD* factorizes our two-way array  $M_{i \times j}$  into three factor matrices containing the left-singular vectors ( $U$ ), the singular values ( $S$ ) and right-singular vectors ( $V$ ) respectively [1, 19]. In case of a reduction to dimension  $k$ , the product ( $M_k$ ) of the resulting matrices represent the best rank- $k$  approximation of our original two-dimensional data structure, picking the  $k$ -largest singular values (refer to Equation 10). Similar to the CP model we can estimate missing values of our original matrix (or rather unknown links in the observed complex network) by multiplication of the single factors:

$$M_k = U_{i \times k} \cdot S_{k \times k} \cdot V_{k \times j}^T \quad (10)$$

Our comparison in Figure 3 reveals that on an average CPES performs better than SVD for **full** datasets. In case of a reduction to the **core** component, CPES achieves almost as good results as SVD. Furthermore we can observe that for both link prediction algorithms the *core* variant is superior. This is due to the fact that *full* networks are more sparse and contain numerous periphery nodes that are not connected with the core. All in all, our proposed multi-way model can compete with existing link prediction algorithms based on matrix factorization techniques, such as matrix *SVD* [1, 6, 17, 19].

## 5 Conclusion and Future Work

This paper presents a novel link prediction approach based on tensor decomposition and exponential smoothing of time series data. In our study we mainly focus on three-dimensional tensors, which capture evolution of social and information networks. For the purpose of analysing temporal trends in evolving networks, we decompose the three-way model into factor matrices; each of them giving information about one separate mode. Trends captured in the “time” factor matrix are used as input for exponential smoothing to extrapolate future points. The extended CP model can be regarded as an approximation of our original tensor, including estimates of future links (refer to Figure 1).

Our proposed CPES link prediction algorithm was tested on various datasets like *Facebook* and *Wikipedia*. As showed in the evaluation part, our approach achieves satisfying results for most of the examined network data, and beyond that reaches precision scores which are comparable to standard matrix-based prediction algorithms. Furthermore we could observe that CPES performs better for preliminary reduced network data, or in other words dense tensors.

The main contribution of this paper is our introduced multi-way model, which incorporates tensor decomposition as well as exponential smoothing techniques to estimate future links in complex networks. As far as we know there does not exist any comparable approach that extends the CP model in a way that it can be used to predict time slices which capture future correlations between nodes

of an evolutionary network.

In future work we want to scrutinize the performance of our CPES algorithm for evolutionary data with more than three dimensions. For instance, acceleration data collected by a mobile phone sensor typically comprise four dimensions, containing gravity information for each of the three axes in an euclidian space as well as a timestamp for each sample. In case of data fusion we could imagine even more than four variables being held in a tensor. We believe that our link prediction algorithm is reliable as long as the data is sufficient, implying that a higher number of dimensions result in more sparse tensors.

## References

1. E. Acar, D. M. Dunlavy, and T. G. Kolda. Link prediction on evolving data using matrix and tensor factorizations. In *ICDMW '09: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, pages 262–269, Washington, DC, USA, 2009. IEEE Computer Society. Ref:13.
2. E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations with missing data. In *SDM10: Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 701–712. SIAM, April 2010. Ref:15.
3. E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Trans. on Knowl. and Data Eng.*, 21(1):6–20, 2009. Ref:11.
4. E. Acar, S. A. amtepe, and B. Yener. Collective sampling and analysis of high order tensors for chatroom communications. In *in ISI 2006: IEEE International Conference on Intelligence and Security Informatics*, pages 213–224. Springer, 2006. Ref:16.
5. B. W. Bader, R. A. Harshman, and T. G. Kolda. Temporal analysis of social networks using three-way dedicom. *Sandia Report*, June 2006. Ref:12.
6. S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285, New York, NY, USA, 1988. ACM.
7. E. Georgii, K. Tsuda, and B. Schölkopf. Multi-way set enumeration in real-valued tensors. In *DMMT '09: Proceedings of the 2nd Workshop on Data Mining using Matrices and Tensors*, pages 1–10, New York, NY, USA, 2009. ACM. Ref:02.
8. S. Janson, D. E. Knuth, T. Luczak, and B. Pittel. The birth of the giant component. *Random Struct. Algorithms*, 4(3):233–359, 1993.
9. T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, June 2008. Ref:14.
10. J. Kunegis, D. Fay, and C. Bauckhage. Network growth and the spectral evolution model. In *Proc. Int. Conf. on Information and Knowledge Management*, 2010.
11. J. Leskovec. Networks, communities and kronecker products. In *CNIKM '09: Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 1–2, New York, NY, USA, 2009. ACM. Ref:04.
12. J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 641–650, New York, NY, USA, 2010. ACM.

13. J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *CHI '10: Conference on Human Factors in Computing Systems*, 2010.
14. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, New York, NY, USA, 2005. ACM. Ref:07.
15. J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 695–704, New York, NY, USA, 2008. ACM.
16. D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, New York, NY, USA, 2003. ACM.
17. D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, New York, NY, USA, 2003. ACM.
18. N. Ma, E.-P. Lim, V.-A. Nguyen, A. Sun, and H. Liu. Trust relationship prediction using online product review data. In *CNIKM '09: Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 47–54, New York, NY, USA, 2009. ACM.
19. S. Spiegel, J. Kunegis, and F. Li. Hydra: a hybrid recommender system [cross-linked rating and content information]. In *CNIKM '09: Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 75–80, New York, NY, USA, 2009. ACM.
20. S. H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2001.
21. J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, New York, NY, USA, 2006. ACM. Ref:09.
22. P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50, New York, NY, USA, 2008. ACM. Ref:10.