Charlier et al., Int J Comput Softw Eng 2018, 3: 132
https://doi.org/10.15344/2456-4451/2018/132

International Journal of
Computer & Software Engineering

Research Article Open Access

Non-Negative Paratuck2 Tensor Decomposition Combined to LSTM Network for Smart Contracts Profiling

Jeremy Charlier* **and Radu State**

SEDAN, University of Luxembourg, 29 Avenue J.F Kennedy, L-1855, Luxembourg, Luxembourg

Abstract

Background: Past few months have seen the rise of blockchain and cryptocurrencies. In this context, the Ethereum platform, an open-source blockchain-based platform using Ether cryptocurrency, has been designed to use smart contracts programs. These are self-executing blockchain contracts. Due to their high volume of transactions, analyzing their behavior is very challenging. We address this challenge in our paper.

Methods: We develop for this purpose an innovative approach based on the non-negative tensor decomposition Paratuck2 combined with long short-term memory. The objective is to assess if predictive analysis can forecast smart contracts activities over time. Three statistical tests are performed on the predictive analytics, the mean absolute percentage error, the mean directional accuracy and the Jaccard distance.

Results: Among dozens of GB of transactions, the Paratuck2 tensor decomposition allows asymmetric modeling of the smart contracts. Furthermore, it highlights time dependent latent groups. The latent activities are modeled by the long short term memory network for predictive analytics. The highly accurate predictions underline the accuracy of the method and show that blockchain activities are not pure randomness.

Conclusion: Herein, we are able to detect the most active contracts, and predict their behavior. In the context of future regulations, our approach opens new perspective for monitoring blockchain activities.

Publication History:

Received: February 15, 2018
Accepted: April 05, 2018
Published: April 07, 2018

Keywords:

Linear Algebra Algorithms, Neural Networks, Blockchain Predictive Analytics

Introduction

In the next months, public institutions and governments will certainly start regulating the non-regulated activities of cryptocurrencies such as Bitcoin or Ether. Some governments already claimed they were investigating cryptocurrencies activities [1-3]. These regulations will probably introduce new sets of rules and ask for more transparency among the blockchain players. As a result, financial products would probably require key information document to advise potential investors of the risk of these investments. Ethereum, with already more than one million accounts, is one of the major platforms for smart contracts relying on Ether cryptocurrency for its existence. Still, the platform supports very few documentation about how blockchain players interact. It also lacks of transparency for non-specialists. Modeling smart contracts and predictive analytics is thus essential for future regulation purpose. Our contributions are twofolds:

1. We describe Paratuck2 Tensor Decomposition (TD) for smart contracts. A non-negative scheme is presented to determine a set of latent factors, where a huge multi-dimensional matrix is decomposed into a less dimensional structure.
2. A second contribution is the prediction of smart contracts activities using Long Short Term Memory (LSTM) trained on Paratuck2 TD. The main novelty is the prediction of future activities by using a set of latent factors. We used LSTM since it has been shown to learn from both long term and recent observations.

We describe the theoretical foundations of our approach in the section materials and methods. We explain how our approach is applied to smart contracts profiling. Then, the results and discussion section highlights the predictions of Ethereum smart contracts exchanges over time. Finally, we conclude with pointers to future works.

Materials and Methods

The first tensor decomposition, the Candecomp/Parafac has been introduced in [4] and [5]. It has been followed then by more complex decomposition [6] that were used for large scale latent analysis [7], [8]. The Paratuck2 decomposition, introduced in [9], was used by Bro in [10] for food analysis and in [11,12] for signal analysis.

In parallel, LSTM networks were introduced in [13] to solve the problem of vanishing gradients of Recurrent Neural Networks (RNN) [14]. It has opened a wide range of applications domains for predictive analytics, space analytics and trajectories modeling [15-17].

Consequently, we present the theoretical tensor background, and then the novel non-negative scheme for the Paratuck2 resolution. Finally, we explain how to perform latent predictions with LSTM.

Mathematical Foundations

Terminology in this paper follows the one described by Kolda and Bader in [18]. Scalars are denoted by lower case letters, a. Vectors and matrices are described by boldface lowercase letters and boldface capital letters, respectively $\mathbf{a}$ and $\mathbf{A}$. High order tensors are represented using upper case letter notation such as X.

*Corresponding Author: Jeremy Charlier, SEDAN, University of Luxembourg, 29 Avenue J.F Kennedy, L-1855, Luxembourg, Luxembourg; E-mail: jeremy.charlier@uni.lu

Citation: Charlier J, State R (2018) Non-Negative Paratuck2 Tensor Decomposition Combined to LSTM Network for Smart Contracts Profiling. Int J Comput Softw Eng 3: 132. doi: https://doi.org/10.15344/2456-4451/2018/132

Citation: Charlier J, State R (2018) Non-Negative Paratuck2 Tensor Decomposition Combined to LSTM Network for Smart Contracts Profiling. Int J Comput Softw Eng 3: 132. doi: https://doi.org/10.15344/2456-4451/2018/132

The transpose matrix of A ∈ $R^{I\times J}$ is denoted by A^T.

The Moore-Penrose inverse of a matrix A ∈ $R^{I\times J}$ is denoted by $A^{\dagger}$

X is called a n-way tensor if X is a n-th multidimensional array. It is expressed by $\mathrm{X} \in \mathrm{R}^{I_1\times I_2\times I_3\times \ldots \times I_n}$.

The square root of the sum of all tensor entries squared of the tensor X defines its norm.

$$\|X\| = \sqrt{\sum_{j=1}^{I_1}\sum_{j=2}^{I_2}\cdots\sum_{j=n}^{I_n} x^2_{j_1,j_2,\ldots,j_n}} \tag{1}$$

The rank-R of a tensor $\mathrm{X} \in R^{I_1\times I_2\times \ldots \times I_N}$ is the number of linear components that could fit X exactly such that

$$\mathrm{X} = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ \ldots \circ a_r^{(N)} \tag{2}$$

with the symbol o representing the vector outer product.

The Kronecker product between two matrices A ∈ $R^{I\times J}$ and B ∈ $R^{K\times L}$, denoted by $A\otimes B$, results in a matrix C ∈ $R^{IK\times KL}$.

$$C = A\otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{bmatrix} \tag{3}$$

The Khatri-Rao product between two matrices A ∈ $R^{I\times K}$ and B ∈ $R^{J\times K}$, denoted by $A\odot B$, results in a matrix C of size $R^{IJ\times K}$. It is the column-wise Kronecker product.

$$C = A\odot B = \begin{bmatrix} a_1\otimes b_1 & a_2\otimes b_2 & \cdots & a_K\otimes b_K \end{bmatrix} \tag{4}$$

Non-Negative Paratuck2

The Paratuck2 decomposition, [9], is well suited for the analysis of intrinsically asymmetric relationships between two different sets of objects. It represents a tensor X ∈ $R^{I\times J\times K}$ as a product of matrices and tensors.

$$\mathrm{X} = \sum_{k=1}^{K} AD_k^A H D_k^B B^T \tag{5}$$

A, H and B are matrices of size $R^{I\times P}$, $R^{P\times Q}$, and $R^{J\times Q}$. The matrices $D_k^A \in R^{P\times P}$ and $D_k^A \in R^{Q\times Q} \forall k \in \{1,\ldots,K\}$ are the slices of the tensors D^A ∈ $R^{P\times P\times K}$ and D^B ∈ $R^{Q\times Q\times K}$. The latent factors p and q are related to the rank of each object set as illustrated in figure 1. The columns of the matrices A and B represent respectively the latent factors p and q, the matrix H describes the asymmetry between the p latent factors and the q latent factors. Finally, the tensors D^A and D^B represent the degree of participation, also called strength, for each of the latent factors, respectively p and q, according to the third dimension.

To achieve the computation of the Paratuck2 decomposition, the following minimization equation has to be solved

$$\min_{\hat{\mathrm{x}}} \|\mathrm{x} - \hat{\mathrm{x}}\| \tag{6}$$

with $\hat{X}$ the approximate tensor described by the decomposition and X the original tensor.

To solve equation 6, the Alternating Least Squares (ALS) method is used as presented by Bro in [10]. All of the matrices and the tensors are updated iteratively. To simplify the resolution explanation, we consider one level k of K, the third dimension of the tensor.

To update A, equation 5 is rearranged such that

$$X_k = AF_k \text{ with } F_k = D_k^A H D_k^B B^T \tag{7}$$

The simultaneous least square solution for all k leads to

$$A = X(F^{\dagger})^T \text{ with } \begin{cases} X = [X_1 X_2 \ldots X_k] \\ F = [F_1 F_2 \ldots F_k] \end{cases} \tag{8}$$

To update D^A, equation 5 is rearranged such that

$$X_k = AD_k^A F_k^T \text{ with } F_k = BD_k^B H^T \tag{9}$$

The matrix D_k^A is a diagonal matrix which lead to the below resolution.

$$D^A_{(k,:)} = [(F_k \odot A) x_k]^T \text{ with } x_k = \mathrm{vec}(X_k) \tag{10}$$

The notation (k, :) represents the *k*-th row of $D^A_{(k,:)}$.

To update H, equation 5 is rearranged such that

$$x_k = (BD_k^B \otimes AD_k^A)\mathrm{h} \text{ with } \begin{cases} x_k = \mathrm{vec}(X_k) \\ \mathrm{h} = \mathrm{vec(H)} \end{cases} \tag{11}$$

which brings the solution

$$h = Z^{\dagger} x \text{ with } Z = \begin{pmatrix} BD_1^B \otimes AD_1^A \\ BD_2^B \otimes AD_2^A \\ \vdots \\ BD_k^B \otimes AD_k^A \end{pmatrix} \tag{12}$$

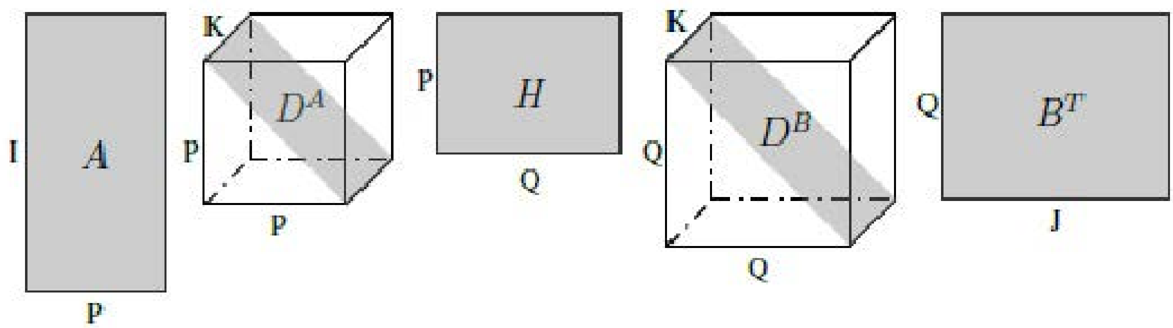

Figure 1: Paratuck2 decomposition of a three-way tensor with dimension notations.

Citation: Charlier J, State R (2018) Non-Negative Paratuck2 Tensor Decomposition Combined to LSTM Network for Smart Contracts Profiling. Int J Comput Softw Eng 3: 132. doi: https://doi.org/10.15344/2456-4451/2018/132

To update **B** and D^B, the methodology presented for the update of **A** and D^A is applied respectively.

In the experiments, we use the non-negative Paratuck2 decomposition leveraging the non-negative matrix factorization presented by Lee and Seung in [19]. The matrices **A**, **B** and **H**, and the tensors D^A and D^B are computed according to the following multiplicative update rule.

$$\begin{cases} a_{ip} \leftarrow a_{ip} \dfrac{[XF^T]_{ip}}{[A(FF^T)]_{ip}}, F = D^A H D^B B^T \\ d^a_{pp} \leftarrow d^a_{pp} \dfrac{[Z^T X]_{pp}}{[D^A (ZZ^T)]_{pp}}, Z = (BD^B H^T) \odot A \\ h_{pq} \leftarrow h_{pq} \dfrac{[Z^T X]_{pq}}{[H(ZZ^T)]_{pq}}, Z = BD^B \otimes AD^A \\ d^b_{qq} \leftarrow d^b_{qq} \dfrac{[XZ]_{qq}}{[D^B (Z^T Z)]_{qq}}, Z = B \odot (H^T D^A A^T)^T \\ b_{qj} \leftarrow b_{qj} \dfrac{[X^T F^T]_{qj}}{[B(FF^T)]_{qj}}, F = (AD^A H D^B)^T \end{cases} \tag{13}$$

with

$$\begin{cases} X = [X_1 X_2 \ldots X_k] \\ X = \mathrm{vec}(x) \end{cases} \tag{14}$$

The non-negative multiplicative update rule helps to better calibration of LTSM since it uses the elements of the tensor decomposition as a starting point. Hereinafter is the complete algorithm of the non-negative ALS Paratuck2 resolution.

Algorithm 1 Non-Negative ALS Paratuck2 with *P* and *Q* latent components for a tensor X of size I × J × K

1. procedure NN-PARATUCK2(X, P , Q)
2. random initialization $A \in R^{I \times P}$, $H \in R^{P \times Q}$, $B \in R^{J \times Q}$
3. set $D_k^A \in R^{P \times P}$ and $D_k^B \in R^{Q \times Q}$ equal to 1 for k = 1,...,K
4. $X = [X_1\ X_2 \ldots X_k]$
5. x = vec(X)
6. repeat:
7. $a_{ip} \leftarrow a_{ip} \dfrac{[XF^T]_{ip}}{[A(FF^T)]_{ip}}, F = D^A H D^B B^T$
8. $d^a_{pp} \leftarrow d^a_{pp} \dfrac{[Z^T X]_{pp}}{[D^A (ZZ^T)]_{pp}}, Z = (BD^B H^T) \odot A$
9. $h_{pq} \leftarrow h_{pq} \dfrac{[Z^T X]_{pq}}{[H(ZZ^T)]_{pq}}, Z = BD^B \otimes AD^A$
10. $d^b_{qq} \leftarrow d^b_{qq} \dfrac{[XZ]_{qq}}{[D^B (Z^T Z)]_{qq}}, Z = B \odot (H^T D^A A^T)^T$
11. $b_{qj} \leftarrow b_{qj} \dfrac{[X^T F^T]_{qj}}{[B(FF^T)]_{qj}}, F = (AD^A H D^B)^T$
12. until maximum number of iterations or stopping criteria satisfied

Latent LSTM predictions

Based on the notation of Sak et al. in [20], LSTM contains memory blocks in the recurrent hidden layer. Each memory block is connected to an input gate and an output gate. Similarly to RNN, the input gate plays the role of the input activation of the memory cells. The output gate is in charge of the flow of cell activations into the rest of the network. In addition, a forget gate is added to the memory block

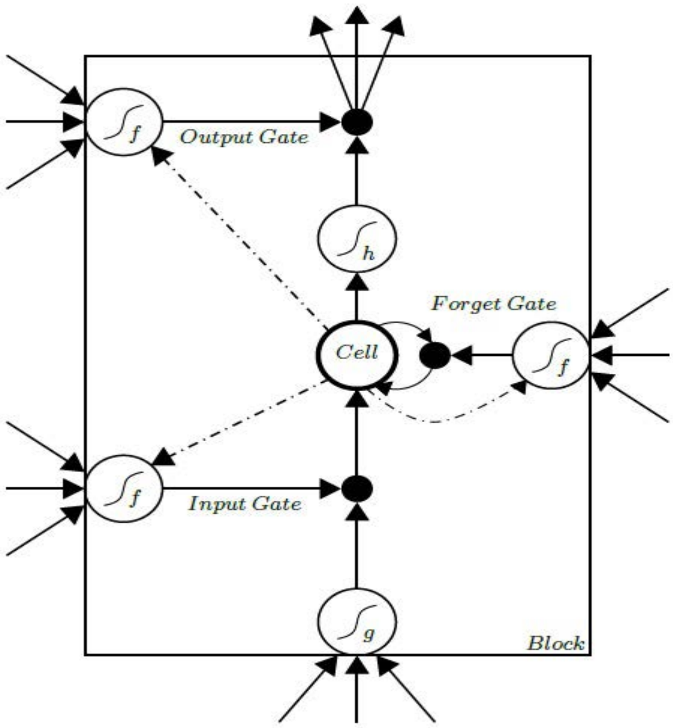

Figure 2: Overview of a LSTM memory cell. In our model, the activation functions *g* and *h* are described by *tanh*, and *f* is the forget gate.

Citation: Charlier J, State R (2018) Non-Negative Paratuck2 Tensor Decomposition Combined to LSTM Network for Smart Contracts Profiling. Int J Comput Softw Eng 3: 132. doi: https://doi.org/10.15344/2456-4451/2018/132

since Gers, Cummins and Schmidhuber presented it in [21]. The forget gate allows the reset of the cell's memory depending on the information received through the input gate. If we consider the input sequence denoted by x such as $x = (x_1,...,x_T)$, the output sequence denoted by y such as $y = (y_1,...,y_T)$ for a sequence of events from t = 1 to t = T. The mapping between x and y for all network unit activations within LSTM is described by the set of equations (15). The activation of the input gate is denoted by i_t, the candidate value for the states of the memory cells by $\tilde{C}_t$, the activation of the memory cells forget gates by f_t, the memory cells new state by C_t, the value of their output gates by o_t and the outputs of the output gates by h_t.

$$\begin{cases} i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ \tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \\ o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \\ h_t = o_t * \tanh(C_t) \end{cases} \tag{15}$$

In the set of equations 15 at time t, x_t stands for the memory cell layer, W_k and U_k with k= {i, c, f, o} for the weight matrices and b_k for the bias vectors. In the model used for the experiments, the activation of a cell's output gate is independent of the memory cell's state C_t such that $V_0 = 0$. The main advantage by fixing $V_0 = 0$ is the ability to perform faster computation, especially on large datasets.

With regards to figure 1, the tensors D^A and D^B collects data about the tensor factorization related to the third dimension, which is very often the time. It means that the evolution of each groups, or clusters, characterized by the latent factors P and Q of the TD contained in the tensors D^A and D^B can be modeled using LSTM. More precisely, LSTM is calibrated on the historical data of the tensors D^A and D^B to predict afterwards the future evolution of each P and Q groups contained in the tensors D^A and D^B as illustrated in figure 3.

Only the diagonals of the tensors D^m with m = {A, B} contain numbers. Therefore, the tensors $D^m \in \mathbb{R}^{L \times L \times K}$ can be reduced to a matrix, $E \in \mathbb{R}^{L \times K}$. The notation L = {P,Q} denotes the latent factors of the Paratuck2 TD.

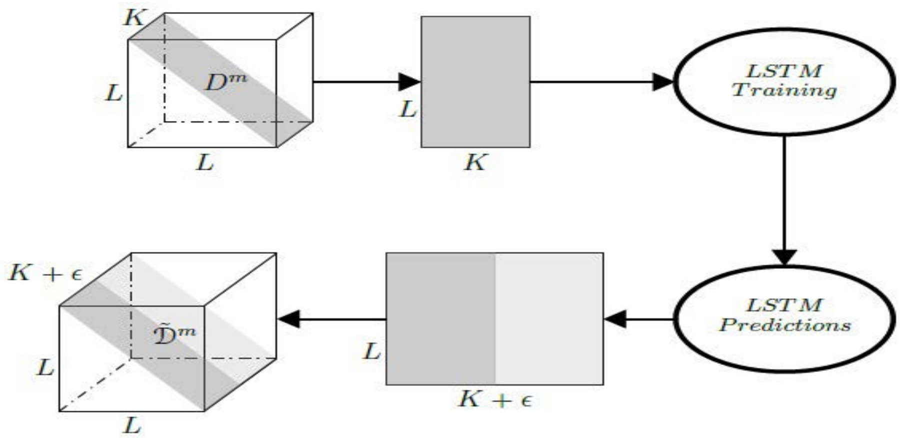

Figure 3: Overview of LSTM training and predictions on the tensor $D^m \in \mathbb{R}^{L \times L \times K}$ with m= {A,B} and L= {P,Q}.

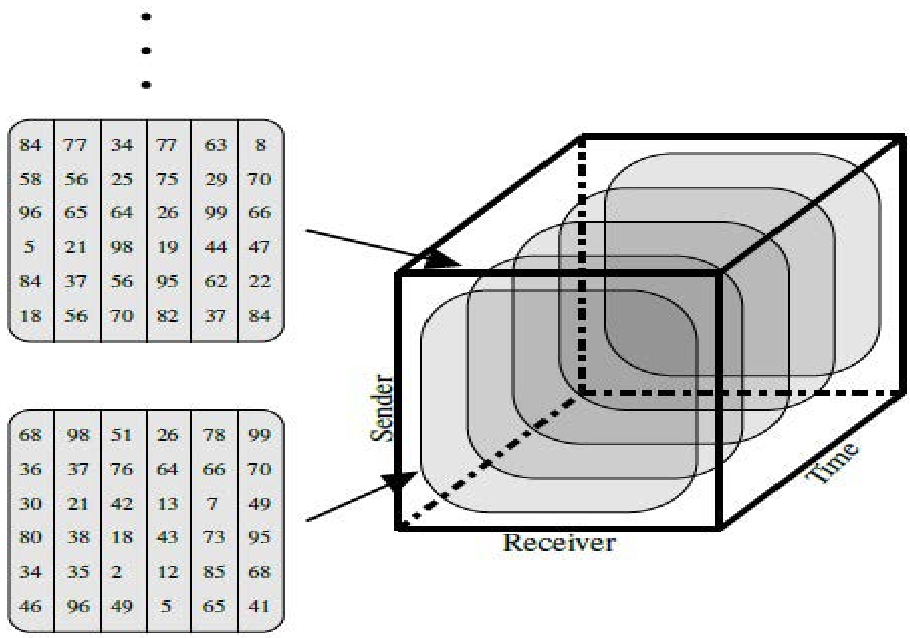

Figure 4: Three way tensor containing Ether amount exchanged between different smart contracts

Citation: Charlier J, State R (2018) Non-Negative Paratuck2 Tensor Decomposition Combined to LSTM Network for Smart Contracts Profiling. Int J Comput Softw Eng 3: 132. doi: https://doi.org/10.15344/2456-4451/2018/132

Data contained in E is then used to train LSTM before performing the predictions on an interval ∈ related to the third dimension K. The resulting matrix of size $R^{L\times(K+\epsilon)}$ gathers the historical data of each latent component L as well as the predicted values. A new tensor denoted by $\tilde{D}_m$ of size RL×L×(K+∈) is built. The methodology is applied on both tensors D^A and D^B for the same ∈. Consequently, the Paratuck2 TD is linked to historical data and predicted data.

Results and Discussion

In this section, we apply our multidisciplinary tensor neural network approach, Paratuck2-LSTM, for Ethereum smart contracts profiling. The experiment is performed on a machine with 15 Intel Xeon E5-4650 v4 2.20 Ghz CPU cores and 80 GB of RAM. We have implemented in Python the algorithm for non-negative Paratuck2 decomposition combined with LSTM code available in [22].

Application to Smart Contracts

Smart contracts activities have been extracted from the Ethereum platform. The data was collected starting 1 January 2016 and ending 1 July 2016. Through the collection process, different data types have been stored, such as the hash key, the sender accounts, the receiver accounts or the blockheights. For the considered six months period, more than 5 millions of transactions have been made. This accounts for an average of 26 transactions per sender account and 18 transactions per receiver account.

In the data set, some smart contracts only relate to one transaction, payment or reception. Such behavior is difficult to predict, and should be considered as unexpected behavior. Our aim is to predict future interactions based on exchanges that already happened. Consequently, only the 1% most active smart contracts have been kept in the training set for their regular activities. This represents a list of 100 smart contracts sending an Ether amount, and a list of 200 smart contracts receiving an Ether amount from the sender contracts.

The features extracted from the dataset are well suited for a tensor representation. Two tensors denoted by $X \in R^{I\times J\times K}$ are built from the Ethereum data. The first dimension, *I*, lists the sender accounts, the second dimension *J*, the receiver accounts and the third dimension, *K*, the time. For each tensor, the interaction between a sender and a receiver is represented by the amount of Ether exchanged at a time. The dense tensor is built based on figure 4. The size of the tensor is $R^{100\times200\times50}$. The tensor is decomposed to highlights the latent component over time. Then, LSTM latent predictions are performed.

As illustrated in figure 5, the information evolving over time is contained in the tensors D^m with m= {A, B}. The matrix **A** gathers static information regarding P senders groups and the matrix **B** static information regarding Q receivers groups. The matrix **H** contains the asymmetric information between the P and the Q latent factors which have been set to respectively to 20 and 30. As a result, the LSTM network is trained on D^m for the sender and the receiver activities predictions.

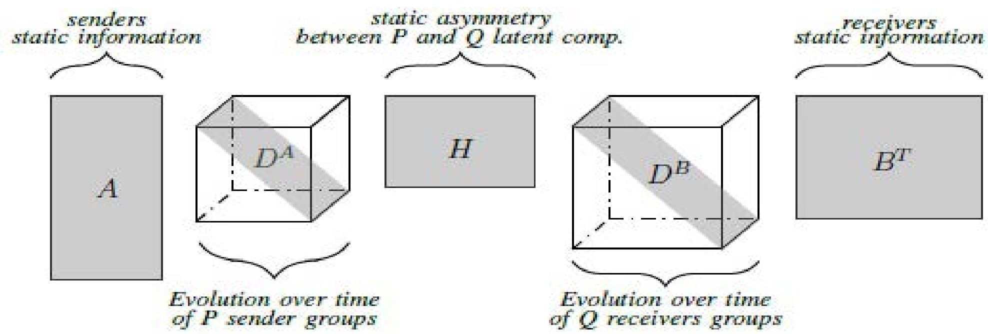

Figure 5: Paratuck2 decomposition applied to smart contracts profiling. The model training and predictions are applied on the tensors D^A and D^B.

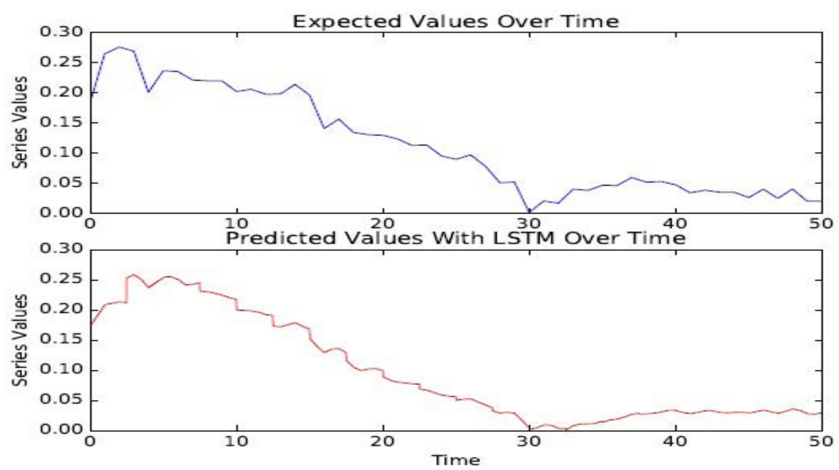

Figure 6: Paratuck2-LSTM applied to smart contracts profiling. This simulation highlights training and predictions on one latent component of the tensor DA.

Predictions results

Figures 6 and 7 show the difference between the true experimental data and the predictions for one rank of the tensors D^A and D^B. The LSTM predictions of smart contracts activities are close to the one observed in the tensor decomposition of the complete true dataset. It means LSTM is appropriate for the modeling of the smart contracts having regular exchanges. To further quantify the accuracy of LSTM predictions, statistical tests are performed. The mean absolute percentage error (MAPE) and the mean directional accuracy (MDA) are computed betwen the predictions and the true data set. A third measure, the Jaccard distance, is also evaluated. As a benchmark for LSTM predictions, the results are compared to the predictions performed by a Decision Tree (DT).

Tables 1 and 2 highlights similar MAPE results for both LSTM predictions and DT predictions. Differences are not significant. On the other hand, the MDA score is lot higher, around a factor 7, for LSTM predictions than for DT predictions. It means the LSTM predictions are able to better reproduce the variations observed in the smart contracts activities through time than the DT predictions. From these first statistical tests, we can observe that the LSTM model is able to reproduce the changes over time of smart contracts activities. It outperforms the decision tree benchmarking algorithm. In addition, the Jaccard distance is computed to underline the distribution divergence between the predictions and the true experimental data. In table 3, it can be observed that LSTM predictions are significantly closer to the true experimental distribution than DT predictions. All LSTM Jaccard distances are within the range 0.20 and 0.30 while the DT Jaccard

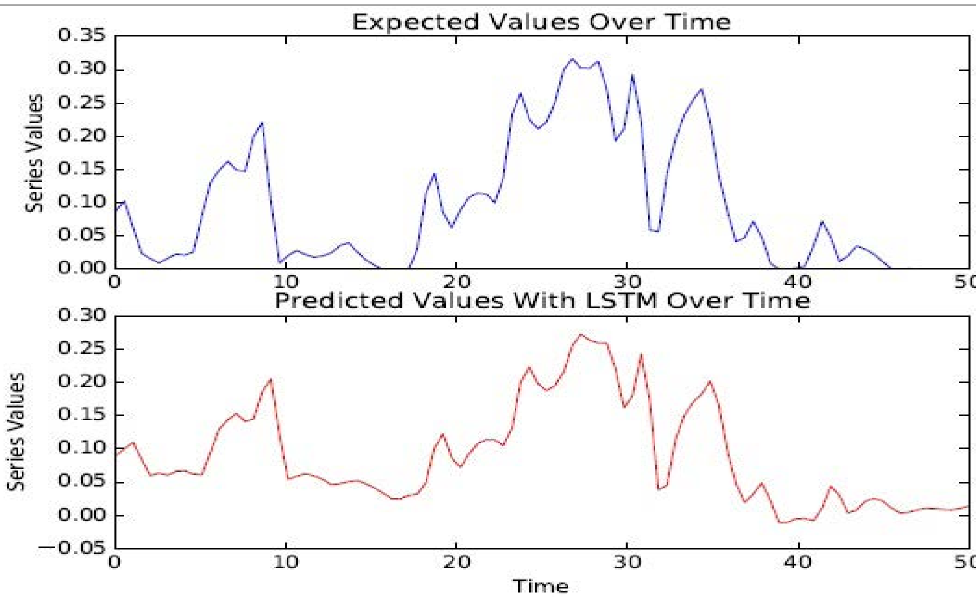

Figure 7: Paratuck2-LSTM applied to smart contracts profiling. This simulation highlights training and predictions on one latent component of the tensor D^B.

Test	MAPE LSTM	MDA LSTM	MAPE DT	MDA DT
1 lat. var.	0.0189	0.5800	0.0092	0.0800
All lat. var.	0.0206	0.5955	0.0112	0.0785

Table 1: Tests to assess the accuracy of LSTM predections on tensor D^A. Predactions are challenged by a decision tree (DT).

Test	MAPE LSTM	MDA LSTM	MAPE DT	MDA DT
1 lat. var.	0.0285	0.5800	0.0320	0.0800
All lat. var.	0.0239	0.6283	0.0188	0.0795

Table 2: Tests to assess the accuracy of LSTM predections on tensor D^B. Predactions are challenged by a decision tree (DT).

Test	Jaccard Dist. LSTM	Jaccard Dist. DT
1 D^A lat. var.	0.2091	0.5725
All D^A lat. var.	0.2453	0.5900
1 D^B lat. var.	0.2942	0.6866
All D^B lat. var.	0.3015	0.6639

Table 3: Jaccard distances to assess the accuracy of LSTM predactions. Predactions are challenged by a decision tree (DT).

distances are between 0.57 and 0.69. LSTM Jaccard distances are between 2 to 3 times lower than the DT Jaccard distances. It confirms the MDA scores in tables 1 and 2.

From the highlighted results, the combined approach of Paratuck2-LSTM delivered good results, validated visually and statistically. It outperformed the DT benchmarking for predictive analytics on several statistical criteria including the MDA and the Jaccard distance.

Conclusion

We proposed in this paper a multi-disciplinary approach leveraging multidimensional linear algebra and neural networks for modeling the complex activities occurring on a certain type of blockchains. Our method combines Paratuck2 tensor decomposition and LSTM to predict behavior in relation to asymmetric data over time. The asymmetry is expressed within the tensor decomposition using two sets of latent factors related to two sets of objects. Our use case considered sender and receiver contracts of the Ethereum platform. Our approach allowed to detect common behaviors over time. Furthermore, it was able to predict accurate interactions and exchanges. We validated our results using statistical tests.

Although the method showed good results in terms of accuracy, it currently lacks the required scalability to be used on big data sets. This is due to the non-negative ALS update rule which is time and memory consuming. We plan to address in future works this issue and develop additional resolution method to the Paratuck2 tensor decomposition using other iterative schemes. Last but not least, the better scalability of the method would help to increase the accuracy of the LSTM network as the training could be performed on longer time period and smaller time step discretization. We plan to address a particular use-case about fraud detection and detection of suspicious behavior over time.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgements

The authors would like to thank Beltran Borja Fiz Pontiveros for his help in Ethereum data manipulation.

References

1. Helms K (2017) Indias government seeks public comments on how bitcoin should be regulated. Bitcoin News.
2. Woolf N (2016) Why the US government wants to bring cryptocurrency out of the shadows. The Guardian.
3. Rees T (2017) Regulating bitcoin: how new frameworks could be a catalyst for cryptocurrencies. The Telegraph.
4. Harshman RA (1970) Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis.
5. Carroll JD, Chang JJ (1970) Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. Psychometrika, 35: 283-319.
6. Harshman RA (1978) Models for analysis of asymmetrical relationships among n objects or stimuli. In Paper presented at the First Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology, Hamilton, Ontario.
7. Harshman RA, Lundy ME (1992) Three-way dedicom: Analyzing multiple matrices of asymmetric relationships. In Annual Meeting of the North American Psychometric Society.
8. Bader BW, Harshman RA, Kolda TG (2007) Temporal analysis of semantic graphs using ASALSAN. Seventh IEEE International Conference.
9. Harshman RA, Lundy ME (1996) Uniqueness proof for a family of models sharing features of tucker's three-mode factor analysis and parafac/candecomp. Psychometrika 61: 133-154.
10. Bro R (2012) Multi-way analysis in the food industry: models, algorithms, and applications.
11. Kibangou A, Favier G (2007) Blind joint identification and equalization of wiener-hammerstein communication channels using paratuck- 2 tensor decomposition. In Signal Processing Conference, 15th European.
12. Rui W, Sheng RG, Yang Z, Peng X (2016) A novel semi-blind receiver algorithm based on paratuck2 model in mimo relay communication system. International Journal of Future Computer and Communication 5: 214.
13. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9: 1735-1780.
14. Cho K, Merrienboer BV, Gulcehre C, Bahdanau D, Bougares F, et al. (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv.
15. Malhotra P, Vig L, Shroff G, Agarwal P (2015) Long short term memory networks for anomaly detection in time series.
16. Greff K, Srivastava RK, Koutnik J, Steunebrink BR, Schmidhuber J, et al. (2017) Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems.
17. Alahi A, Goel K, Ramanathan V, Robicquet A, Fei-Fei L, et al. (2016) Social lstm: Human trajectory prediction in crowded spaces.
18. Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM review 51: 455-500.
19. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401: 788-791.
20. Sak H, Senior A, Beaufays F (2014) Long short-term memory recurrent neural network architectures for large scale acoustic modeling.
21. Gers FA, Schmidhuber J, Cummins F (1999) Learning to forget: Continual prediction with lstm.
22. Chollet F (2015) Keras.