Non-Negative PARATUCK2 Tensor Decomposition
Combined to LSTM Network For Smart Contracts Profiling

Jeremy Chartier
SEDAN, SnT
University of Luxembourg
Luxembourg, Luxembourg
jeremy.charlier@uni.lu

Radu State
SEDAN, SnT
University of Luxembourg
Luxembourg, Luxembourg
radu.state@uni.lu

Jean Hilger
Information Technology
BCEE
Luxembourg, Luxembourg
j.hilger@bcee.lu

Abstract—Smart contracts are programs stored and executed on a blockchain. The Ethereum platform, an open-source blockchain-based platform, has been designed to use these programs offering secured protocols and transaction costs reduction. The Ethereum Virtual Machine performs smart contracts runs, where the execution of each contract is limited to the amount of gas required to execute the operations described in the code. Each gas unit must be paid using Ether, the crypto-currency of the platform. Due to smart contracts interactions evolving over time, analyzing the behavior of smart contracts is very challenging. We address this challenge in our paper. We develop for this purpose an innovative approach based on the non-negative tensor decomposition PARATUCK2 combined with long-short term memory (LSTM) to assess if predictive analysis can forecast smart contracts interactions over time. To validate our methodology, we report results for two use cases. The main use case is related to analyzing smart contracts and allows shedding some light into the complex interactions among smart contracts. In order to show the generality of our method on other use cases, we also report its performance on video on demand demonstration.

Keywords – PARATUCK2 Tensor Decomposition; LSTM; Predictive Analytics

I. INTRODUCTION

In the next few months, public institutions and governments would probably start regulating the non-regulated activities of cryptocurrencies such as Bitcoin or Ether as some of the governments already claimed they were investigating cryptocurrencies activities ([1], [2] and [3]). These regulations would probably introduce new sets of rules and would ask for more transparency among the market players. As a result, financial products would probably require key information document (KID) 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 and lacks of transparency for non-specialists. Modeling smart contracts and predictive analytics is thus essential for future regulation purpose. Our contributions are twofolds:

- We describe a PARATUCK2 Tensor Decomposition (TD) for smart contracts. This decomposition leads to a a set of latent factors, where a huge multi-dimensional matrix is decomposed into a less-dimensional structure.

- A second contribution is the prediction of smart contracts activities using LSTM trained on PARATUCK2 TD. The main novelty is that we predict future activities among a huge quantity of entities by predicting the evolution of a set of latent factors using LSTM. We used LSTM since this approach has been shown to learn from both long term experience and recent observations. Once the latent factors are predicted, we reconstruct the multidimensional matrices using the predicted latent factors.

To outline our approach, we present the related work of tensor decomposition, Long-Short-Term Memory (LSTM) and smart contracts in the first section of the paper. Section 2 provides the fundamentals of the chosen tensor decomposition (TD) and how LSTM are applied to the decomposition. Section 3 is made from two distinct parts. The first part illustrates the concepts of the approach on a small example on Video On Demand (VOD) for recommender systems (RS). The second part applies the methodology to the profiling and the predictions of Ethereum smart contracts exchanges among time. Finally, we conclude with pointers to future works.

II. RELATED WORK

In this section, we briefly mention the background of DEDICOM TD to its further extension called PARATUCK2. We also underlined related work in smart contracts and LSTM.

A. PARATUCK2 Tensor Decomposition Background

Harshman, Caroll and Chang have been the first to introduce the concept of the multidimensional tensors and their resolutions with CANDECOMP/PARAFAC (CP) tensor decomposition in [4] and [5]. Since then, CP decomposition has been used in various applications and domains. In [6], Giudici and Pecora proposed to use it for the analysis of international bilateral claims. It underlined existing communities, followed their evolution over time and major event such as the 2008 banking crisis. In [7], the authors applied a novel compression-based CP decomposition to hyperspectral imaging and hyperspectral data.

Furthermore, other decompositions have been introduced afterward such as the DEDICOM decomposition by Harshman in [8]. Unlike the CP decomposition, it offers the possibility to illustrate asymmetric relationships among data. Harshman and Lundy in [9] illustrated the concepts of the DEDICOM...
decomposition by applying the method to international trade data between different nations. In [10], Bader, Harshman, and Kolda proposed a new algorithm for DEDICOM’s resolution applied on Enron’s email to analyze communities before Enron’s crisis.

For the analysis of interactions between two possibly different sets of objects, Harshman and Lundy introduced the PARATUCK2 decomposition in [11]. The algorithm has been applied by Bro in [12] to flow injection analysis system with a constraint on pH-gradients. In the recent years, the decomposition has been also applied in communication networks. In [13], Kibangou and Favier considered a special design of an input signal in communication channels to show that the output signal could be represented using PARATUCK2 representation. The authors in [14] proposed a novel algorithm for its resolution applied to Multiple Input Multiple Output (MIMO) relay communication for faster convergence. However, the drawback of PARATUCK2 is that the decomposition results in a complex factorization leading to non-trivial analysis which could explain why it has been less commonly applied among scientific community.

B. LSTM as one of the state-of-the-art neural network

The problem of the vanishing gradients in Recurrent Neural Networks (RNN) was solved in an elegant way in [15] by Hochreiter and Schmidhuber through the LSTM architecture, which subsequent found multiple applications in a broad area of applications that range from speech recognition and hand writing to social media analysis. For a comprehensive overview on the application domains, the reader is referred to [16], where the authors describe eight of the most common LSTM variants on different domains.

The authors in [17] illustrated the efficiency of LSTM in social science by training a model on human trajectories in crowded environment. Once trained, the authors researched if the network was able to predict future large movement of persons and to avoid collisions between humans. The modeling and the predictions of pure time series have been showed in [18] with the study of the resilience of LSTM. In their results, the authors highlighted for various domains such as space shuttle or power demand that a trained network on non-anomalous data could identify anomalies for unknown length of time series.

Based on these previous results, illustrating the efficiency and the scalability of LSTM for time series forecasting, we consider in this paper the extent to which the behaviors of smart contracts can be identified and predicted using LSTM.

C. Emergence Of Smart Contracts On Distributed Ledgers

Introduced in 1994 by Nick Szabo as "a computerized transaction protocol that executes the terms of a contract", a smart contract allows public execution across a distributed networks of nodes through a democratic organization. This is done without requiring a central gatekeeper. We witness in the present an evolution towards real world application domains and business by major industry players. As such, in [19], Frantz and Nowostawski proposed a modeling approach of smart contracts that could transpose human-readable contract into computational equivalents for the contracts execution on the blockchain. The goal of the mechanism is to allow the specification and the interpretation of the smart contracts to larger audience than just blockchain specialists. In [20], Christidis and Devetsikiotis tried to find some applications of the blockchain into Internet of Things (IoT). Although most of the research is optimistic about broader applications of smart contracts, others raised concerns about security. More specifically, in [21], Atzei, Baroletti and Cimoli tested the smart contracts implementation in Ethereum against attacks with the goal of stealing or tampering assets transferred through smart contracts. In their experiments, they showed how a series of attack could lead to a money robbery or to a denial of service attack against the Ethereum blockchain. The cited papers illustrate a common underlying approach: most of the works either extend the use of the smart contracts or highlight the security concerns due to their usage.

In our approach, we propose an innovative application of the PARATUCK2 tensor decomposition for the analysis of smart contracts exchanges. We also combine in a novel approach the PARATUCK2 tensor decomposition with LSTM inherited from neural networks. We show that our method can accurately predict future probable exchanges between smart contracts.

III. PARATUCK2 AND LSTM NETWORKS

In this section, we present in the beginning mathematical operations involved in PARATUCK2 TD followed by the PARATUCK2 description itself. Finally, LSTM for time prediction is described.

A. Tensor Description

**Notation** Terminology in this paper follows the one described by Kolda and Bader in [22] and commonly used by other publications. 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 Euler script notation such as $\mathcal{X}$.

The transpose matrix of $\mathbf{A} \in \mathbb{R}^{I \times J}$ is denoted by $\mathbf{A}^T$ and results in a matrix of size $\mathbb{R}^{J \times I}$.

The Moore-Penrose inverse of a matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ is denoted by $\mathbf{A}^+$ and results in a matrix of size $\mathbb{R}^{J \times I}$.

**Tensor Definition** $\mathcal{X}$ is called a $n$-way tensor if $\mathcal{X}$ is a $n$-th multidimensional array and can be expressed as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n}$.

**Tensor Operations** The square root of the sum of all tensor entries squared of the tensor $\mathcal{X}$ defines its norm.

$$||\mathcal{X}|| = \sqrt{\sum_{j_1=1}^{I_1} \sum_{j_2=1}^{I_2} \cdots \sum_{j_n=1}^{I_n} x_{j_1,j_2,\ldots,j_n}^2}$$  (1)

The rank-$R$ of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is the number of
linear components that could fit $\mathcal{X}$ exactly such that
\[ \mathcal{X} = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ \ldots \circ a_r^{(N)} \]  
(2)
with the symbol $\circ$ representing the vector outer product.

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

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

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

\[ C = A \odot B = [a_1 \otimes b_1 \quad a_2 \otimes b_2 \quad \cdots \quad a_K \otimes b_K] \]  
(4)

B. PARATUCK2 Tensor Decomposition

In our approach, we use the PARATUCK2 decomposition introduced by Harshman and Lundy in [11]. This decomposition is well suited for the analysis of intrinsically asymmetric relationships between two different sets of objects. It represents a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ as a product of matrices and tensors.

\[ \mathcal{X}_k = AD_k^AHD_k^B B^T \quad \text{with} \quad k = \{1, \ldots, K\} \]  
(5)
The matrices $A$, $H$ and $B$ are matrices of size $I \times P$, $P \times Q$ and $I \times Q$, respectively. The matrices $D_k^A \in \mathbb{R}^{P \times P}$ and $D_k^B \in \mathbb{R}^{Q \times Q}$ for $k \in \{1, \ldots, K\}$ are the slices of the tensors $D^A \in \mathbb{R}^{P \times P \times K}$ and $D^B \in \mathbb{R}^{Q \times Q \times K}$. 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 the latent factors $P$ and $Q$, respectively. The matrix $H$ describes the asymmetry between the $P$ latent factors and the $Q$ latent factors. Finally, the tensors $D^A$ and $D^B$ measure the evolution of the latent factors regarding the third dimension.

To achieve the computation of the PARATUCK2 decomposition, the following minimization equation has to be solved
\[ \min_{\hat{\mathcal{X}}} ||\mathcal{X} - \hat{\mathcal{X}}|| \]  
(6)
with $\hat{\mathcal{X}}$ the approximate tensor described by the decomposition and $\mathcal{X}$ the original tensor.

To solve equation (6) the Alternating Least Squares (ALS) method is used as presented by Bro in [12]. 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
\[ \mathcal{X}_k = AF_k \quad \text{with} \quad F_k = D_k^AHD_k^B B^T \]  
(7)
The simultaneous least square solution for all $k$ leads to
\[ A = X(F^T)^T \quad \text{with} \quad X = [X_1 \quad X_2 \cdots \quad X_k] \]  
(8)
To update $D^A$, equation (5) is rearranged such that
\[ X_k = AD_k^A F_k^T \quad \text{with} \quad F_k = BD_k^B H^T \]  
(9)
The matrix $D_k^A$ is a diagonal matrix which lead to the below resolution.
\[ D_k^{A(k,:)} = [(F_k \odot A)x_k]^T \quad \text{with} \quad x_k = \text{vec}(X_k) \]  
(10)
The notation $(k,:)$ represents the $k$th row of $D_k^{A(k,:)}$.

To update $H$, equation (5) is rearranged such that
\[ x_k = (BD_k^B \odot AD_k^A)h \quad \text{with} \quad [x_k \quad h] = \text{vec}(H) \]  
(11)
which brings the solution
\[ h = Z'x \quad \text{with} \quad Z = \begin{bmatrix} (BD_k^B \odot AD_k^A) \\ (BD_k^B \odot AD_k^A) \\ \vdots \\ (BD_k^B \odot AD_k^A) \end{bmatrix} \]  
(12)
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 [23]. The matrices $A$, $B$ and $H$, and the tensors $D^A$ and $D^B$ are computed according to the following multiplicative update rule.

\[
\begin{align*}
    a_{ip} & \leftarrow a_{ip} \frac{[XF^T]_{ip}}{[A(FF^T)]_{ip}} \\
    d_{pp}^a & \leftarrow d_{pp}^a \frac{[Z'x]_{pp}}{[D^A(ZZ^T)]_{pp}} \\
    h_{pq} & \leftarrow h_{pq} \frac{[Z'x]_{pq}}{[H(ZZ^T)]_{pq}} \\
    d_{qq}^b & \leftarrow d_{qq}^b \frac{[D^B(Z'Z)]_{qq}}{[B(FF^T)]_{qq}} \\
    b_{qj} & \leftarrow b_{qj} \frac{X^T[F^T]_{qj}}{[B(FF^T)]_{qj}} \\
\end{align*}
\]
(13)
with
\[
\begin{align*}
    X &= [X_1 \quad X_2 \cdots \quad X_k] \\
    x &= \text{vec}(X) 
\end{align*}
\]  
(14)
The multiplicative update rule helps to better calibration of LSTM that uses the elements of the tensor decomposition as a starting point.

C. LSTM applied to PARATUCK2

Our contribution contains besides an application of the non-negative PARATUCK2 scheme to smart contracts, also a computational step based on LSTM. LSTM has been introduced in [15] by Hochreiter and Schmidhuber with the goal of overcoming the short-ends of RNN. As described by the authors in [24], a RNN is a neural network with a hidden state, \( h \), and an optional output \( y \). It operates on an event sequence \( x \). RNN suffers from the vanishing gradient which leads to the loss of long-term memory. The loss of the long-term memory was solved by the LSTM architecture.

Following the notation of Sak, Senor and Beaufays in [25], 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 since Gers, Cummins and Schmidhuber presented it in [26]. The forget gate allows the reset of the cell’s memory since \( Gers, Cummins and Schmidhuber presented it in [26] \).

In the set of equations \( 15 \) at time \( t \), \( x_t \) stands for the memory cell layer, \( W_k \) and \( U_k \) with \( k \in \{i, c, f, o\} \) 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 \( m \in \{A, B\} \) contain numbers which means that the tensors \( D^m \in \mathbb{R}^{L \times L \times K} \) can be reduced to a matrix, \( E \), of size \( \mathbb{R}^{L \times K} \). The notation \( L \in \{P, Q\} \) denotes the latent factors of the PARATUCK2 TD. Data contained in \( E \) is then used to train LSTM before performing the predictions on an interval of size \( \epsilon \) related to the third dimension \( K \). The resulting matrix of size \( \mathbb{R}^{L \times (K+\epsilon)} \) gathers the historical data of each latent component \( L \) as well as the predicted values. A new tensor denoted by \( D^m \) of size \( \mathbb{R}^{L \times L \times (K+\epsilon)} \) is then built. The methodology is applied on both tensors \( D^A \) and \( D^B \) for the same \( \epsilon \) which leads afterward to a PARATUCK2 tensor decomposition linked to historical data as well as predicted data.

IV. PARATUCK2-LSTM: A MULTIDISCIPLINARY TENSOR-RNN BASED APPROACH

In this section, we consider two application domains: a small example based on Recommender Systems (RS) for Video On Demand (VOD) such as Netflix and a larger experiment on the Ethereum platform with smart contracts profiling.

The first experiment is performed on a PC with Intel Core i7 CPU and 8 GB of RAM. The second experiment is more CPU consuming due to the larger size of the data set and thus uses a more powerful machine with 15 Intel Xeon E5-4650 v4 2.20 Ghz CPU cores and 80 GB of RAM. We have implemented in the Python programming language the algorithm for non-
negative PARATUCK2 decomposition combined with LSTM network.

A. A first approach of PARATUCK2-LSTM concepts using recommender systems

In this first example, the objective is to illustrate the main concepts of the proposed method through a simple application. We aim at familiarizing the reader with the understanding of latent factors, matrices and tensors involved in the PARATUCK2 tensor decomposition and how these are combined with LSTM. Hereinafter, the ability of the model to provide personalized movie recommendations is addressed.

A RS is a methodology that analyzes a large volume of dynamically generated information to provide users with personalized content and services ([27]).

The data for this experiment was downloaded from the Movielens data base provided by the University of Minnesota which contains more than 20 millions ratings for more than 135,000 users and 27,000 movies. Since our aim is to provide a small but illustrative example, the dataset used for the experiments has been shortened to 100 users and 125 movies randomly selected from 10 April 2014 to 30 March 2015. The data set is divided in two data sets, one for training from 10 April 2014 to 10 November 2014, and one for simulation and validation that ranges from 11 November 2014 to 30 March 2015. The second dataset allows to cross-validate the prediction simulation with the events that effectively happened and assess the accuracy of the simulation.

Two three-way tensors, denoted by $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, are used for the experiments. The first dimension of $\mathcal{X}$, denoted by $I$, represents the users, the second dimension, $J$, models the movies and the third dimension, $K$, represents the time. Each tensor is generated by counting how many times a user watched a specific movie within a particular time period. The first tensor of size $\mathbb{R}^{I \times J \times K}$ is completed with the first dataset. After the tensor decomposition is performed, LSTM is trained between 10 April 2014 and 10 November 2014. We use the LSTM to perform predictions for a period between 11 November 2014 and 30 March 2015. The second tensor is filled with the complete data set and decomposed. With the use of the two tensors, the results of the complete dataset can be compared with the results simulated by LSTM.

As shown in the figure, LSTM is trained on $\mathcal{X}^A$ and $\mathcal{X}^B$ for users and movies predictions respectively. The matrices $A$, $H$ and $B$ gather static information about the users, the movies and the asymmetry between the P and Q latent factors which have been set to 20 and 30. Static information defines information which does not evolve over the third dimension, the time.

According to figures 5 and 6, the LSTM predictions seem close to the true values. To objectively assess the accuracy of the predictions, the Mean Absolute Percentage Error (MAE) and the mean directional accuracy (MDA) have been evaluated. The MAE is a measure of the prediction of the accuracy for forecasting methods based on the average percentage of
difference between the true series and the predicted series. The MDA is a measure of the predictions of the accuracy based on the sign of the evolution of the time series. It is the average of similar signs in the evolution of the time series between the true values and the predicted values.

For two random selections of one component of P and one component of Q, respectively related to $D^A$ and $D^B$, MAE and MDA criteria are satisfying. In table I, MAE is between 8% and 12% and MDA is between 78% and 80%. In addition, it means that the PARATUCK2-LSTM scheme could be used to observe and predict future evolution of P users clusters and Q movies clusters.

To conclude, this small example illustrates the concepts underlying a PARATUCK2-LSTM analysis. A secondary outcome consists in very good experimental results for series predictions based on the latent factors. These factors, issued through the tensor decomposition, can be assimilated to different ranks or clusters. This leads naturally to an applied approach for observing the evolution over time of each latent factor. Furthermore, the recommender engine could target the recommendation of future movies ranks to future users ranks based on the evolution of $D^m$ with $m = \{A, B\}$.

### B. Smart Contracts profiling using PARATUCK2-LSTM

Hyperledger and Ethereum are the two main blockchains for the use of smart contracts. The data from Ethereum 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, 5.5 millions of transactions have been made. This accounts for an average of 26 transactions per sender accounts and 18 transactions per receiver accounts.

As LSTM is trained on historical data, it is well suited to reproduce events that already happened but can hardly simulate events that never happened. For such tasks, stochastic process models are better choices. 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 more like unexpected behavior. For fraud detection, LSTM could be thus appropriate. Since our aim is to predict future interactions based on exchanges that already happened in the past and not to detect suspicious behavior, only the 1% most active smart contracts have been kept in the training set due to their regular activities.

The complete dataset is divided into two datasets, one collecting events between the 1 January 2016 and the 1 April 2016 and another one between 2 April 2016 and 1 July 2016. The split of the dataset allows to cross-validate the results between the predictions of exchanges performed by LSTM and the true exchanges performed between the smart contracts.

Following the methodology described in the first experiment for RS, two tensors denoted by $X \in \mathbb{R}^{I \times J \times K}$ 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 (figure 7). The first tensor is built and decomposed with the first part of the dataset resulting in a tensor size of $\mathbb{R}^{100 \times 200 \times 25}$. LSTM is trained over the time period of the first dataset and used to perform predictions for the period between 2 April 2016 and 1 July 2016. The second tensor is filled directly with the complete dataset and decomposed. The second tensor has a size of $\mathbb{R}^{100 \times 200 \times 50}$.

As illustrated in figure 8, 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 exchange predictions.

According to figures 9 and 10 which show the difference between the true data and the predictions for one rank of the tensors $D^A$ and $D^B$, the predictions of Ether exchanges are close to the one observed in the tensor decomposition of the
complete true dataset. It means that LSTM can be seen as appropriate for the modeling of the smart contracts having regular exchanges. MAE and MDA have been computed to assess objectively the difference between the simulated results and the true data.

Table II illustrates the accuracy of the results obtained from the interactions predictions using LSTM. If we consider only one latent factor selected randomly among $P$ and $Q$, respectively for $D_A$ and $D_B$, the MAE is between 5% and 12% while the MDA is around 72%. If we consider the average of all latent factors $P$ and $Q$, the MAE is between 78% and 79% and the MDA is between 75% and 81%. From statistical point of view, the values for the overall MAE and overall MDA are accurate. From the highlighted results of this application, the combined approach of PARATUCK2-LSTM delivers good results validated visually and statistically.

V. CONCLUSIONS

We proposed in this paper a multi-disciplinary approach leveraging multidimensional linear algebra and neural net-

Table II

| Test           | Values for $D_A$ | Values for $D_B$ |
|----------------|------------------|------------------|
| 1 latent var. MAE | 0.7239          | 0.5829           |
| 1 latent var. MDA | 0.7174          | 0.7166           |
| Average MAE     | 0.0791          | 0.0776           |
| Average MDA     | 0.7534          | 0.8073           |
works for modeling the complex interactions occurring on a certain type of blockchains. Our method combines both of the 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. We instantiated this model on two use cases. The first practical case was applied on users and movies as concrete objects. The second use case considered sender and receiver contracts of the Ethereum platform. Our approach allowed to detect in both cases common behaviors over time and was able to predict accurate recommendations in the first case and accurate interactions and exchanges in the second experiment. 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 on particular use-case about fraud detection and detection of suspicious behavior over time.

ACKNOWLEDGMENT

The authors would like to thank Beltran Borja Fiz Pontiveros for his help in Ethereum data manipulation. They also thank Jacques Putz and Patric de Waha from Banque et Caisse d’Épargne de l’État (BCEE) as one of the strongest support for the research publication.

REFERENCES

[1] Kevin Helms. Indias government seeks public comments on how bitcoin should be regulated. Bitcoin News, 2017.
[2] Nicky Woolf. Why the us government wants to bring cryptocurrency out of the shadows. The Guardian, 2016.
[3] Tom Rees. Regulating bitcoin: how new frameworks could be a catalyst for cryptocurrencies. The Telegraph, 2017.
[4] Richard A Harshman. The parafac procedure: models and conditions for an “explanatory” multimodal factor analysis. 1970.
[5] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. Psychometrika, 35(3):283–319, 1970.
[6] Paolo Giudici, Nicolò Pecora, and Alessandro Spelta. Tensorial factor analysis of international bilateral claims. 2016.
[7] Miguel A Veganzones, Jeremy E Cohen, Rodrigo Cabral Farias, Jocelyn Chanussot, and Pierre Comon. Nonnegative tensor cp decomposition of hyperspectral data. IEEE Transactions on Geoscience and Remote Sensing, 54(5):2577–2588, 2016.
[8] Richard A Harshman. 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, August, 1978.
[9] RA Harshman and ME Lundy. Three-way dedicom: Analyzing multiple matrices of asymmetric relationships. In Annual Meeting of the North American Psychometric Society, 1992.
[10] Brett W Bader, Richard A Harshman, and Tamara G Kolda. Temporal analysis of semantic graphs using asalsan. In Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, pages 33–42. IEEE, 2007.
[11] Richard A Harshman and Margaret E Lundy. Uniqueness proof for a family of models sharing features of tucker’s three-mode factor analysis and parafac/candecomp. Psychometrika, 61(1):133–154, 1996.
[12] Rasmus Bro. Multi-way analysis in the food industry: models, algorithms, and applications. PhD thesis, Københavns Universitet, LUKKET: 2012 Det Biovidenskabelige Fakultet for Fødevarer, Veterinærmedicin og Naturres-sourcereFaculty of Life Sciences, LUKKET: 2012 Institut for FødevarervidenskabDepartment of Food Science, LUKKET: 2012 Kvalitet og TeknologiQuality & Technology, 1998.
[13] Alain Kibangou and Gérard Favier. Blind joint identification and equalization of wiener-hammerstein communication channels using paratuck-2 tensor decomposition. In Signal Processing Conference, 2007 15th European, pages 1516–1520. IEEE, 2007.
[14] Wang Rui, Rui Guo-Sheng, Zhang Yang, and Xue Peng. A novel semi-blind receiver algorithm based on paratuck2 model in mimo relay communication system. International Journal of Future Computer and Communication, 5(6):214–216, 2016.
[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
[16] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems, 2017.
[17] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. pages 961–971, 2016.
[18] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. page 89, 2015.
[19] Christopher K Frantz and Mariusz Nowostawski. From institutions to code: towards automated generation of smart contracts. In Foundations and Applications of Self* Systems, IEEE International Workshops on., pages 210–215. IEEE, 2016.
[20] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. IEEE Access, 4:2292–2303, 2016.
[21] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sox). In International Conference on Principles of Security and Trust, pages 164–186. Springer, 2017.
[22] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. SIAM review, 51(3):455–500, 2009.
[23] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755):788–791, 1999.
[24] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.
[25] Hasım Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.
[26] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. Ijmm, 1999.
[27] FO Isinkaye, YO Folajimi, and BA Ojokoh. Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3):261–273, 2015.