Neuron Mathematical Model Representation of Neural Tensor Network for RDF Knowledge Base Completion

Farhad Abedini [a], Mohammad Bagher Menhaj [b,*], Mohammad Reza Keyvanpour [c]

[a] Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[b] Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran
[c] Department of Computer Engineering, Alzahra University, Vanak, Tehran, Iran

Received 25 November 2015; revised 3 August 2016; accepted 10 October 2016; available online 16 March 2017

Abstract

In this paper, a state-of-the-art neuron mathematical model of neural tensor network (NTN) is proposed to RDF knowledge base completion problem. One of the difficulties with the parameter of the network is that representation of its neuron mathematical model is not possible. For this reason, a new representation of this network is suggested that solves this difficulty. In the representation, the NTN is modeled as a multi-layer perceptron (MLP) network and the tensor parameter can be distributed into the new network neurons. Moreover, it is suggested that the inputs can be converted into one vector rather than the inputs of NTN are two correlated vectors at the same time. The results approve that the NTN does not indeed represent a new neural network and the implementation results easily confirm it can be considered as another representation of the MLP network. So, the first idea is representation of a neuron based mathematical model for the NTN through the ordinary and yet well-defined neural network concepts and next contribution will be equivalency proof of the two NTN and suggested MLP networks.

Keywords: *Semantic Web, Knowledgebase Completion, Neural Tensor Network, Multi-Layer Perceptron Network, RDF Data Model.*

1. Introduction

RDF is a standard data model for data interchange on the semantic web [1]. Semantic web is a web of linked data [2] and the linked data has been implemented by RDF data model [1]. For reasoning data in the semantic web a comprehensive knowledge base is necessary [3]. The flow of RDF knowledge base has been shown in Fig. 1.

Fig. 1. The flow of RDF knowledge base

However there is no any comprehensive RDF knowledge base in the web. For this reason, fact extraction methods have been developed until the knowledge base is comprehensive [3-28]. In Fig. 2 it has been illustrated that fact extraction can be done from available knowledge base such as [3-12] and from other resources same as [13-28]. Fact extraction from available resource can be considered as knowledge base enrichment [25-28]. There are two methods for the enrichment but the focuse of this paper is on the RDF knowledge base completion.

* Corresponding author. Email: menhaj@aut.ac.ir

RDF knowledge bases are structured of a set of facts, each of which is called a triple. Every fact, is contained two entities and a relation between them. The first entity is called subject and the second one is object, while their relation is called predicate. For example, the fact that "Obama was born in Hawaii" can be shown as the triple of <Obama, born-in, Hawaii>, in which the Obama is the first entity, Hawaii is the second entity and born-in is the relationship between them. RDF knowledge bases are diverse including YAGO [3], DBpedia [6], Freebase [7], and WordNet [8].

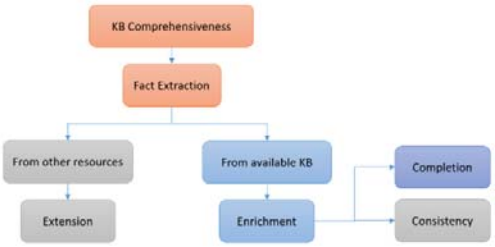

Fig. 2. Structure of resolving incomprehensiveness problem

Since RDF knowledge bases have been presented for realization of the semantic Web, they should be comprehensive. However, unfortunately there is no fully comprehensive knowledge base, where their extension and enrichment is a big concern of the current knowledge bases as can be observed in Fig. 2. To enrich knowledge bases two problems of incompletion and inconsistency must be solved. But, the focus of this paper is the incompletion.

So far, numerous methods have been proposed for completion of RDF knowledge bases including [20-24] and for RDF knowledge base enrichment such as [25-28]. In the completion methods new facts is reasoned from available facts of knowledge base. But the methods of knowledge base extension extract new facts from an other resources. Since extraction of facts out of unstructured sources can be very demanding, methods such as SOFIE [5] have been propounded which use the facts available in the knowledge base itself to complete and enrichment their facts?

In the SOFIE method, in order to enrich YAGO ontology, a method has been suggested that is able to extract new facts out of unstructured texts. In this method, the accuracy of the facts gained from the unstructured texts is computed by the facts available in the knowledge base. However, in [21], using reasoning in the neural tensor network (NTN) by the facts available in the knowledge base, a method has been proposed for completion of the RDF knowledge base. In comparison to other associated methods, this method gives better results and is able to increase RDF facts to the knowledge base. In fact, so far this method is state-of-the-art in the area of knowledge base completion using available knowledge base [21].

One of the problems of the NTN is that representation of its neuron mathematical model is not possible because of presence of the tensor parameter. To solve this problem, in this paper a neural network has been proposed that is equivalent to NTN in which its neuron mathematical model can be represented. For this purpose, it is suggested that a multi-layer perceptron network (MLP) be used instead of the NTN. Using an MLP network, the network is easily represent able, where it is possible to use the learning rules of MLP networks for it. Indeed, in this paper it has been demonstrated that the NTN is not a new neural network and its neuron mathematical model is also proposed. So, contributions of this paper are as follows:

- Representation of a neuron based mathematical model for the NTN through the ordinary and yet well-defined neural network concepts
- Equivalency proof of the two NTN and suggested MLP networks
- New solution for the RDF knowledgebase completion

In continue, first the NTN is introduced. Then, NTN neuron mathematical model will be represented. For this aim, first, neuron mathematical model of tensor layer is represented and then, NTN will be presented as a two-layer perceptron network. Next, the NTN and MLP are trained using NTN learning rules and implementation results show that both of them are equivalent. Finally, the results will be discussed.

2. Review of the NTN

In this section, a review of neural tensor network (NTN)is presented. NTN was proposed by Socher et al. for RDF knowledge base completion that is state-of-the-art [21]. In this section, this network model is introduced that reasons over knowledge bases by learning vector representations over them. Each relation triple <*e1*, R, *e2*> is described by a neural network and pairs of entities*e1* and *e2* which are given as input to that relation's model. If the entities are correlated then the network returns a high score otherwise a low one. Because of this, any triple can be scored with some certainty mentioned implicitly or explicitly in the knowledge base.

The aim of the model is to state whether two entities *e1* and *e2* are in certain relation R. For example, whether in triple <*Obama, born-in, Hawaii*> the *born-in* relation between two entities *Obama* and *Hawaii* is true and with what score of certainty. It is supposed that vector representations of the two entities be $e1, e2 \in R^d$.

Indeed, the NTN network replaces a standard linear neural network layer with a bilinear layer that directly relates the two entity vectors across multiple dimensions. By this model, a score of how probable it is that two entities are in certain relationship can be computed. If *d* dimension vectors($e1, e2 \in R^d$) be representation of two entities then, the function of NTN that scores the relationship of them can be demonstrate as equation (1) [21].

$$g(\mathrm{e1, R, e2}) = \mathrm{U^T}\, f\left(\mathrm{e1^T W_R^{[1:k]} e2} + \mathrm{V_R}\begin{bmatrix}\mathrm{e1}\\ \mathrm{e2}\end{bmatrix} + \mathrm{b_R}\right) \quad (1)$$

Where $f = tanh$ is a standard nonlinearity applied element-wise. $W_R^{[1:k]} \in R^{d\times d\times k}$ is a tensor and the bilinear tensor product $e1^T W_R^{[1:k]}\, e2$ results in a vector $h \in R^k$, where each entry is computed by one slice $i = 1...k$ of the tensor: $h_i = e1^T W_R^{[1:k]} e2$. The other parameters for relation R are the standard form of a neural network: $V_R \in R^{k\times 2d}$, $U_R \in R^d$ and $b_R \in R^d$ [21]. Details of parameters are presented in Fig. 3in which the Eq. (1) has been expanded. In this figure, value of *d* has been considered 3 dimensions as sample.

Its basic excellence is that it can correlate the two vectors multiplicatively as inputs instead of only implicitly through the nonlinearity as with standard neural networks where the entity vectors are simply concatenated [21].

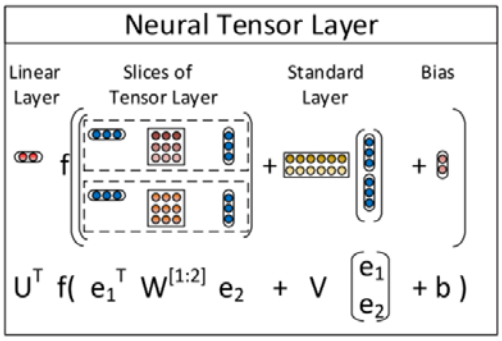

Fig. 3. Neural Tensor Network Model [21]

The main idea is that each triple in the training set $T^{(i)} = <e1^{(i)}, R^{(i)}, e2^{(i)}>$ should receive a higher score than a triple in which one of the entities is replaced with a random entity. There are N_R many relations, indexed by $R^{(i)}$ for each triple. Each relation has its associated NTN parameters. We call the triple with a random entity corrupted and denote the corrupted triple as $T_c^{(i)} = <e1^{(i)}, R^{(i)}, e_c>$ where we sampled entity e_c randomly from the set of all entities that can appear at that position in that relation. Let the set of all relationships' NTN parameters be Ω = u, W, V, b, E. Now, objective function in Eq. (2) must be minimized [6].

$$J(\Omega) = \sum_{i=1}^{N}\sum_{c=1}^{C} max(0, 1 - g(T^{(i)}) + g(T_c^{(i)})) + \lambda||\Omega||_2^2 \quad (2)$$

Where N is the number of training triples and the correct relation triple is scored higher than its corrupted one up to a margin of 1. For each correct triple C random corrupted triple is sampled. Standard L_2 regularization of all the parameters, weighted by the hyper parameter λ has been used.

Taking derivatives from this objective function is not possible. For this reason, an optimization method [29] has been used for its minimization. Then the

model is trained by taking derivatives with respect to the five parameters. Therefore, in this paper it is proposed that NTN can be equivalent with a standard neural network. With this suggestion, standard learning rules can be applied for the training. The suggested representation is explained in continue.

3. Suggestion a New Neuron Mathematical Model Representation for the NTN

In this part, a new method for neuron based mathematical model for the NTN through the ordinary and yet well-defined neural network concepts is introduced. By this mathematical model, it is indicated that the NTN is equivalent to a MLP network. For this aim first, neuron based mathematical model for tensor layer is proposed. Next, the NTN is represented as an MLP network and then, it is proved that these two networks are equivalent. Final, in the implementation section these claims are confirmed.

3.1. New Representation of NTN Neuron Mathematical Model with one Tensor Layer

In this section one slice of tensor layer is considered as in Fig. 4 is clear. Since, there are two input vectors e1 and e2in the tensor layer and each of them is located at either sides of the parameter matrix of W, the neuron based mathematical model of this layer cannot be represented directly. For this reason, a new representation for this layer is proposed.

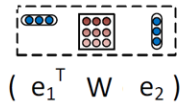

Fig. 4. One slice of tensor layer

In the tensor layer there are two input vectors of *e1* and *e2* that have been multiplied in the parameter matrix of *W*. In Eq. (3) this issue is shown and its results is a scalar number. For easily, the entity vectors is considered as three-dimension which are generalizable to further dimensions. Expansion of this Eq. is in Eq. (4).

$$= [e1_1 e1_2 e1_3] \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} e2_1 \\ e2_2 \\ e2_3 \end{bmatrix} \quad (3)$$

$$= [e1_1 e2_1 w_{11} + e1_2 e2_2 w_{12} + e1_3 e2_3 w_{13} + e1_1 e2_1 w_{21} + e1_2 e2_2 w_{22} + e1_3 e2_3 w_{23} + e1_1 e2_1 w_{31} + e1_2 e2_2 w_{32} e1_3 e2_3 w_{33}] \quad (4)$$

Therefore, if W and P are considered as Eq. (5) and Eq. (6) then the score g from Eq. (1) is gained as (7).

$$W = [w_{11}, w_{21}, w_{31}, w_{12}, w_{22}, w_{32}, w_{13}, w_{23}, w_{33}] \quad (5)$$

$$P = [e1_1 e2_1, e1_2 e2_2, e1_3 e2_3, e1_1 e2_1, e1_2 e2_2, e1_3 e2_3, e1_1 e2_1, e1_2 e2_2, e1_3 e2_3]^T \quad (6)$$

$$g = f(WP) \quad (7)$$

As it is clear in Eq. (7), two input vectors e1 and e2 can be converted into one that is called P. Indeed, it is proposed that one vector can represent the correlation between these two vectors. So, representation of the mathematical model for the NTN tensor layer with the new parameter W can be presented as Fig. 5 for which no bias value has been existed.

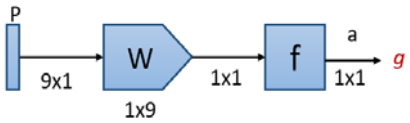

Fig. 5. Neuron mathematical model of NTN with one tensor layer

This model can be used for representation of the NTN as MLP network that is presented in the next part.

3.2. Representation of the NTN in the form of the MLP

In this part for easily, the parameters b and V in the NTN are not considered. To obtain all values for all facts of the training set, one score called g is assigned to each of the facts. To recognize the score of true from incorrect facts, a threshold value must be gained. After that, true and false facts can be classified.

To obtain score of g, in Fig. 6 two slice of tensor layer is considered. Their further detail is in Eq. (8).

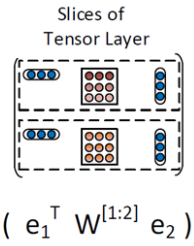

Fig. 6. Two slices of tensor layer

$$h = [e1_1 e1_2 e1_3]\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}\begin{bmatrix} e2_1 \\ e2_2 \\ e2_3 \end{bmatrix} + [e1_1 e1_2 e1_3]\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \end{bmatrix}\begin{bmatrix} e2_1 \\ e2_2 \\ e2_3 \end{bmatrix} \quad (8)$$

If parameter W be in the form of Eq. (9) and by considering the input vector as P in Eq. (10), the score g can be gained as Eq. (11) with parameter U from Eq. (1).

$$W = \begin{bmatrix} w_{11}, & w_{21} & ,w_{31} & ,w_{12} & ,w_{22} & ,w_{32} & ,w_{13} & ,w_{23} & ,w_{33} \\ w'_{11}, & w'_{21}, & w'_{31}, & w'_{12}, & w'_{22}, & w'_{32}, & w'_{13}, & w'_{23}, & w'_{33} \end{bmatrix} \quad (9)$$

$$P = [e1_1 e2_1, e1_2 e2_2, e1_3 e2_3, e1_1 e2_1, e1_2 e2_2, e1_3 e2_3, e1_1 e2_1, e1_2 e2_2, e1_3 e2_3]^T \quad (10)$$

$$g = U^T f(WP) \quad (11)$$

To represent the statement of Eq. (11) in the form of the neuron mathematical model, in Fig. 7 a neuron mathematical model of two layer perceptr on network is proposed. As can be observed, by changing the representation of inputs and the network parameters, an MLP network has been gained.

Parameter U is shown in second layer and the function of this layer is linear.

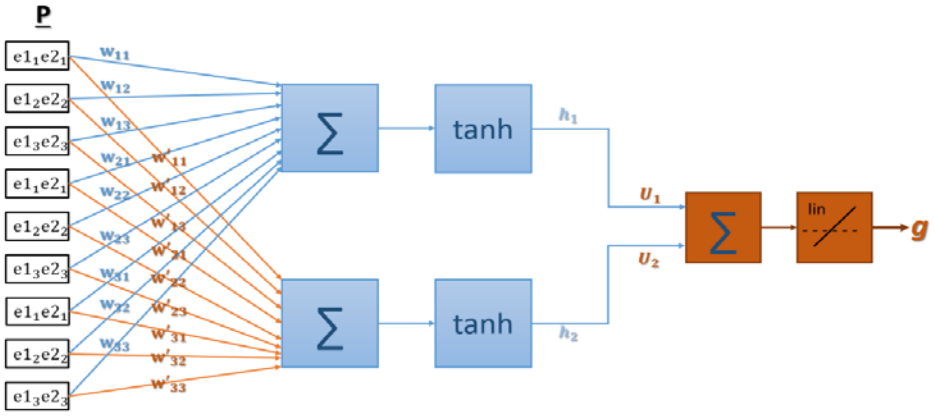

Fig. 7. Suggested network for representation of neuron mathematical model of NTN

The general representation of the suggested model is displayed in Fig. 8. These models will be generalized for further slices as well, signifying that for every slice one neuron can be considered.

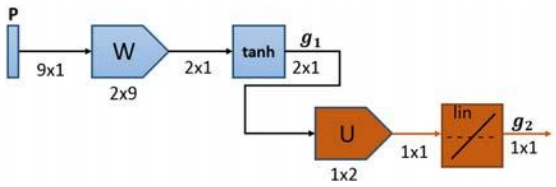

Fig. 8. General representation of the suggested MLP model

The calculation method of the output a in a perceptron network with L layers can be observed in (12).

$$a^0 = P$$
$$a^{l+1} = f^{l+1}(W^{l+1}a^{l+1} + b^{l+1}), \quad l = 0, 1, 2, \ldots, L-1 \quad (12)$$

So, by considering $a= g$, $b=0$, and $L=2$, the g^2 output of the mentioned network can be gained as (13).

$$g^0(P,R) = P$$
$$g^1(P,R) = f^0(W_R g^0) = Tanh(W_R P) = g_1 \quad (13)$$
$$g^2(P,R) = f^1(U_R g^1) = U_R^T Tanh(W_R P) = g_2$$

The result of Eq. (13) is equal to Eq. (11) ($g_2 = g$). Therefore, the suggested network can be equivalent with the NTN. But, for output of the network to be in the range of 1 and -1, *tan hyperbolic* (*tanh*) should be considered instead of the linear one. This issue is displayed in Fig. 9.

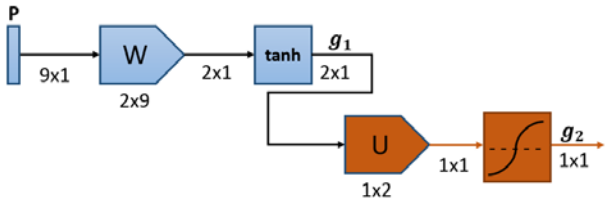

Fig. 9. The suggested MLP model with *tanh* function

In this way, equivalency of the new neuron based mathematical model with the NTN was illustrated. In order to prove this claim, in the Implementation section, the results obtained from the training of these networks are compared.

4. Implementation Results

In previous section, it was proved that NTN is equivalent with an MLP network by mathematical proof. In this section, the implementation results can confirm it. For this aim, each two networks are trained and tested on the benchmark dataset of [21] paper. In this dataset, Word Net is considered as the knowledgebase. Here, the results of these networks are compared on the dataset.

4.1. NTN Training

For NTN training, taking derivatives from the objective function in Eq. (2) is not possible. For this reason, an optimization method has been used for its minimization. This method is min Func [29]. Then the model is trained by taking derivatives with respect to the five groups of parameters of NTN. An abstract view of neural tensor network training algorithm can be shown in algorithm 1.

Algorithm 1: NTN training algorithm

Input: T: Triples with structure <e1,R,e2>, theta: Array of NTN Parameters

Output: Array of Trained Parameters, Cost

1: **for** iteration = 1 : MaxItration

2: selected_data = Select 20000 random train data Triples T

3: data_R = selected_data.R

4: data_e1 = selected_data.e1

5: data_e2 = selected_data.e2

6: data_ec = Select 20000 random entities as Corrupted

7: T = <data_e1, data_R, data_e2>

8: Tc = <data_e1, data_R, data_ec>

9: [theta, cost] = minFunc(J(theta, T, Tc), theta)

10: costs(iteration) = cost

11: **end for**

12: **Return** theta, costs

In this algorithm, the cost function J is used that had been introduced in Eq. (2). This algorithm is trained by taking derivatives with respect to the 5 groups of parameters. Derivative for tensor parameter W is shown in Eq. (14) and derivatives for other parameters are same as the BP.

$$\frac{\partial g(e1,R,e2)}{\partial W} = uTanh'(z)e_1e_2^T, \quad \text{Where} \quad z = e_1^T W e_2 + V\begin{bmatrix}e_1\\e_2\end{bmatrix} + b \tag{14}$$

Input of this algorithm are a set of true triples from knowledge base called “T”, and an array of different entries of all network parameters called “theta”. Here for easiness of training, total entries of all parameters have been placed in single file and have been considered as a unique array (theta). But after training step, all of parameters will be separated. The output is final values of parameters after training and cost of the training. A set of false triples as corrupted triples is called “T_c” in this algorithm.

The function of *minFunc* is a Matlab function for unconstrained optimization of differentiable real-valued multivariate functions using line-search

methods. It uses an interface very similar to the Matlab Optimization Toolbox function *fminunc*, and can be called as a replacement for this function. On many problems, *minFunc* requires fewer function evaluations to converge than *fminunc*. Further it can optimize problems with a much larger number of variables (*fminunc* is restricted to several thousand variables), and uses a line search that is robust to several common function pathologies [29].

The default parameters of *minFunc* call a quasi-Newton strategy, where limited-memory BFGS updates with Shanno-Phua scaling are used in computing the step direction, and a bracketing line-search for a point satisfying the strong Wolfe conditions is used to compute the step direction. In the line search, (safeguarded) cubic interpolation is used to generate trial values, and the method switches to an Armijo back-tracking line search on iterations where the objective function enters a region where the parameters do not produce a real valued output [29].

After implementation of this algorithm in Matlab, the parameters are regularized and the NTN can be used for classification of true and false triples. For analyse of NTN training, diagram of training costs in each iteration is indicated in Fig. 11.

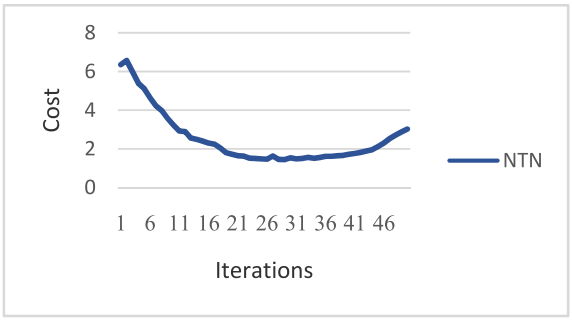

Fig. 11. Diagram of NTN training costs in each iteration

This diagram shows one run of the algorithm. In table 1 min costs of three run of this algorithm are explained. Each run take about 30 min time. These values will be compared to values of next algorithm.

Table. 1. Min-costs of NTN training

	First run	Second run	Third run	Average
Min Cost	1.452	1.4562	1.4437	**1.4506**
#Iteration	25	27	26	**26**

4.2. Training of the Suggested MLP Network

MLP training algorithm is introduced in algorithm 2. Because of changing in structure of NTN, for this algorithm, inputs must be changed. For this aim each two vectors of e1 and e2 is multiplied as P.

Algorithm 2: Training algorithm for the suggested MLP network

Input: T: Triples with structure <e1, R, e2>, theta: Array of NTN Parameters

Output: Array of Trained Parameters, Cost

1: **for** iteration = 1 : MaxItration
2: selected_data = Select 20000 random train data Triples T
3: data_R = selected_data.R
4: data_e1 = selected_data.e1
5: data_e2 = selected_data.e2
6: data_ec = Select 20000 random entities as Corrupted
7: data_P = data_e1 .* data_e2
8: Pc = data_e1 .* data_ec
9: F = <data_P, data_R >
10: Fc = <data_Pc, data_R >
11: [theta, cost] = minFunc(J'(theta, F, Fc), theta)
12: costs(iteration) = cost
13: **end for**
14: **Return** theta, costs

In this algorithm, the cost function J' is used that is introduced in Eq. (15).

$$J'(\Omega) = \sum_{i=1}^{N}\sum_{c=1}^{C} max(0,\ 1 - a(F^{(i)}) + a(F_c^{(i)})) + \lambda||\Omega||_2^2 \quad (15)$$

Where $a = U^T f(WP), F^{(i)} = < P^{(i)}, R^{(i)} >,$ $F_c^{(i)} = < Pc^{(i)}, R^{(i)} >.$

Since available dataset is the same as that of the NTN, a method similar to the NTN can be used for training of the new network. As (1) was derived in relation with each of the parameters in the NTN at every stage, in this method it should be derived in relation with all parameters at every stage with its cost being calculated based on the objective function (14). The derivative with respect to parameter W is

provided in (15) and the derivative with respect to parameter U is provided in (16).

$$\frac{\partial g(P,R)}{\partial W} = U_R^T Tanh'(W_R\, P)\, P \tag{16}$$

$$\frac{\partial g(P,R)}{\partial U} = Tanh(W_R\, P) \tag{17}$$

In order to obtain the cost of calculation of the objective function, first some triples are chosen out of the training dataset, where some entities take the place of the second entity of that fact per each of them. In this way, for every correct triple, some incorrect triples are achieved. Overall, a number of correct and incorrect triples will be achieved as the final training data. The objective function is applied to these data, with the cost of each stage being calculated. Using optimization methods, the network parameters can be altered through optimizing the cost of every stage.

After implementation of this algorithm in Matlab, the parameters are regularized and the MLP network can be used for classification of true and false triples. For analyse of the training algorithm, diagram of training costs in each iteration is indicated in Fig. 12.

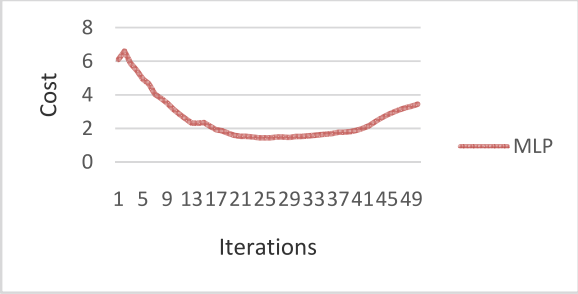

Fig. 12. Diagram of the MLP training costs in each iteration

This diagram shows one run of the algorithm. In table 2 min costs of three run of this algorithm are explained. Each run take about 30 min time. These values will be compared to values of NTN algorithm. This comparison is indicated in Fig. 13.

Table. 2. Min-Costs of two layer perceptron training

	First run	Second run	Third run	Average
Min Cost	1.480	1.4295	1.4409	**1.4501**
#Iteration	24	25	28	**25.66**

Accordingly, the trained network and the final value of parameters are obtained in a way that the objective function cost is reduced. Therefore, with representation of NTN neuron mathematical model, the learning method of the new model was presented that is equivalent with the NTN. Fig. 13 confirms this claim. Each step of two algorithms is the same and parameters values are similar.

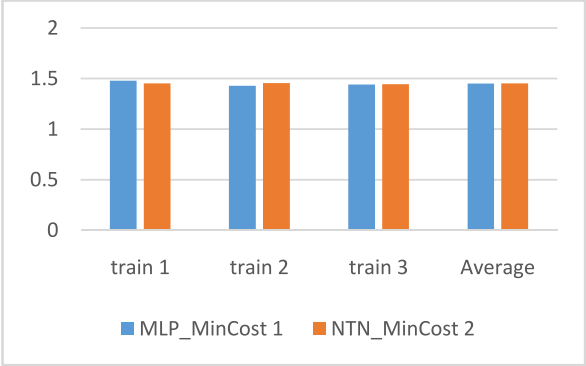

Fig. 13. Comparison of two tables of 2 and 3

After training step, two networks was tested by test data of the benchmark dataset and same results are gained. In test step, some true and false triples is given to trained networks as inputs. After test, each network classify these inputs. The classification of each two networks are similar to each other and accuracy of both is 86.2. So, the implementation results confirm equivalency of two networks.

5. Conclusions

In this paper, it had been shown that the NTN can be represented by an MLP neural network. Indeed, it was illustrated that these two networks are equivalent. For this aim, first NTN was introduced and its problems were studied. Considering that this method is stat-of-the-art, but one of its main disadvantages was disability of its neuron mathematical model representation.

To solve this problem, by changing inputs and parameters structure a new method for representation of neuron mathematical model was proposed. The new network was a two layer perceptron that its

output equation was convertible to NTN ones. So, it is proved that NTN is not a new neural network, even it is another representation of available neural networks. Indeed, the suggested model is a new solution for RDF knowledge base completion problem that has ability of the state-of-the-art.

Overall, the models of this paper were used for the RDF data model that has two correlated entity vectors. In fact, in this paper, the NTN was criticized and it had been shown that correlation of two entity vectors of network input can be considered as one correlated vector with the corresponding parameters. This idea can be used in the future works by finding new correlation methods for pair entities with results improvement. Therefore, this method can be generalized for problems that have two parallel input vectors in which the correlation is important. For future work, it is considered that the network will be trained using standard learning rules of the MLP networks. These learning rules may improve the results.

References

[1] Bizer, Ch.; Heath, T.; Berners-Lee, T., "Linked data-the story so far." Semantic Services, Interoperability and Web Applications: Emerging Concepts pp. 205-227 (2009).

[2] Berners-Lee, T., Hendler, J.; OraLassila. "The semantic web." Scientific American 284.5 pp. 28-37 (2001).

[3] Suchanek, F. M.; Kasneci, G.; Weikum, G., "Yago: a core of semantic knowledge." Proceedings of the 16th international conference on World Wide Web. ACM, (2007).

[4] Biega, J.; Kuzey, E.; Suchanek, F. M.; "Inside YAGO2s: A transparent information extraction architecture." Proceedings of the 22nd International Conference on World Wide Web. ACM, (2013).

[5] Suchanek, F. M.; Sozio, M.; Weikum, G., "SOFIE: a self-organizing framework for information extraction." Proceedings of the 18th international conference on World Wide Web. ACM, (2009).

[6] Lehmann, J.; et al. "DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia." Semantic Web 6.2 pp. 167-195 (2015).

[7] Bollacker, K.; et al. "Freebase: a collaboratively created graph database for structuring human knowledge." Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, (2008).

[8] Miller, G.A. WordNet: A Lexical Database for English. Communications of the ACM, (1995).

[9] Gerber, D.; Hellmann, S.; Bühmann, L.; Soru, T.; Usbeck, R., Ngomo, A. C. N. (2013, October). Real-time RDF extraction from unstructured data streams. In International Semantic Web Conference. Springer Berlin Heidelberg. pp: 135-150 (2013).

[10] Kuzey, E.; Weikum, G. "Extraction of temporal facts and events from Wikipedia." Proceedings of the 2nd Temporal Web Analytics Workshop. ACM, (2012).

[11] Lawrie, D.; et al. "Comparing and Evaluating Semantic Data Automatically Extracted from Text." AAAI 2013 Fall Symposium on Semantics for Big Data. AAAI Press, November (2013).

[12] Seifert, Ch.; et al. "Crowdsourcing fact extraction from scientific literature." Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data. Springer Berlin Heidelberg, pp: 160-172 (2013).

[13] Abedini, F.; Mahmoudi, F.; Jadidinejad, A. H. OFE: Ontological Facts Extraction from text for computing Semantic Relatedness. In proceedings of IEEE 3th International Conference on Machine Learning and Computing (ICMLC 2011), Singapore, 26-28 February pp: 84-88 (2011).

[14] Abedini, F.; Mahmoudi, F.; Jadidinejad, A. H. A New Disambiguation Method for Semantic Entity Extraction Using YAGO Ontology. In proceedings of IEEE 3th International Conference on Machine Learning and Computing (ICMLC 2011), Singapore, 26-28 February pp: 79-83 (2011).

[15] Abedini, F.; Mahmoudi, F.; Jadidinejad, A. H. SESR: Semantic Entity Extraction for computing Semantic Relatedness. In proceedings of 4th International Conference on Advanced Computer Theory and Engineering (ICACTE2011), Dubai, UAE, 28-30 December, pp: 225-228 (2011).

[16] Abedini, F.; Mahmoudi, F.; Jadidinejad, A. H. "From Text to Knowledge: Semantic Entity Extraction using YAGO Ontology," International Journal of Machine Learning and Computing vol. 1, no. 2, pp: 113-119 (2011).

[17] Abedini, F.; Mahmoudi, F.; Mirhashem, S. M. "Using Semantic Entity Extraction Method for a New Application," International Journal of Machine Learning and Computing vol. 2, no. 2, pp: 178-182 (2012).

[18] Abedini, F.; Mirhashem, S. M. "From Text to Facts: Recognizing Ontological Facts for a New Application," International Journal of Machine Learning and Computing vol. 2, no. 3, pp: 183-187 (2012).

[19] Abedini, F.; Mirhashem, S. M. "Entity Disambiguation in Text by YAGO Ontology," International Journal of Computer Theory and Engineering vol. 5, no. 3, pp; 432-435 (2013).

[20] Chen, D.; et al. "Learning new facts from knowledge bases with neural tensor networks and semantic word vectors." arXiv preprint arXiv:1301.3618 (2013).

[21] Socher, R.; et al. "Reasoning with neural tensor networks for knowledge base completion." Advances in Neural Information Processing Systems. (2013).

[22] West, R.; et al. "Knowledge base completion via search-based question answering." Proceedings of the 23rd international conference on World widweb. International World Wide Web Conferences Steering Committee, (2014).

[23] He, W.; et al. "Knowledge Base Completion Using Matrix Factorization." Asia-Pacific Web Conference. Springer International Publishing, (2015).

[24] Zhao, Y.; et al. "Knowledge base completion by learning pairwise-interaction differentiated embeddings." Data Mining and Knowledge Discovery 29.5 pp: 1486-1504 (2015).

[25] Bühmann, L.; Lehmann, J., "Pattern based knowledge base enrichment." The Semantic Web–ISWC 2013. Springer Berlin Heidelberg, pp: 33-48 (2013).

[26] Hellmann, S.; et al. "Knowledge Base Creation, Enrichment and Repair." Linked Open Data--Creating Knowledge Out of Interlinked Data. Springer International Publishing, pp: 45-69 (2014).

[27] Hwang, M.; Dongjin, Ch.; Pankoo, K. "A method for knowledge base enrichment using Wikipedia document information." Information-An International Interdisciplinary Journal 13.5 (2010): 1599-1612.

[28] Bühmann, L.; Lehmann, J. "Universal OWL axiom enrichment for large knowledge bases." Knowledge Engineering and Knowledge Management. Springer Berlin Heidelberg, pp: 57-71 (2012).

[29] Schmidt, M. minfuc http://people.cs.ubc.ca/ schmidtm/software/minfunc.html (2005).