Knowledge-Based Systems 228 (2021) 107258

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Taylor's theorem: A new perspective for neural tensor networks

Wei Li, Luyao Zhu, Erik Cambria *

School of Computer Science and Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 24 December 2020
Received in revised form 20 June 2021
Accepted 24 June 2021
Available online 26 June 2021

Keywords:
Neural tensor networks
Natural language processing
Taylor's theorem
Conversational sentiment analysis

ABSTRACT

Neural tensor networks have been widely used in a large number of natural language processing tasks such as conversational sentiment analysis, named entity recognition and knowledge base completion. However, the mathematical explanation of neural tensor networks remains a challenging problem, due to the bilinear term. According to Taylor's theorem, a kth order differentiable function can be approximated by a kth order Taylor polynomial around a given point. Therefore, we provide a mathematical explanation of neural tensor networks and also reveal the inner link between them and feedforward neural networks from the perspective of Taylor's theorem. In addition, we unify two forms of neural tensor networks into a single framework and present factorization methods to make the neural tensor networks parameter-efficient. Experimental results bring some valuable insights into neural tensor networks.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

A neural tensor network (NTN) explicitly associates two entities and can be applied to many natural language processing (NLP) tasks such as knowledge base completion [1], question answering [2,3], natural language inference [4], word segmentation [5], entity disambiguation [6], semantic compositionality [7] and conversational sentiment analysis [8]. Taking the conversational sentiment analysis task as an example, given two adjacent utterances "The film is terrible" and "Yes, except for the beginning", the first utterance changes the sentiment polarity of the second utterance from neutral to negative. Here, the interactions between two utterances could be captured by NTN.

NTN is composed of three parts, namely the bilinear term, linear term and bias term. The representation of NTN varies from model to model. For example, [7] employs bilinear and linear terms; [6] utilizes only the bilinear term. Moreover, NTN is proved to be more powerful than feedforward neural networks [9]. However, how to explain NTN from the mathematical perspective is still a challenging problem, due to the bilinear term. In this paper, we associate NTN with Taylor's theorem and find that each slice of NTN could be represented as a 2nd order multivariate Taylor polynomial. Moreover, we apply Taylor's theorem to feedforward neural networks and thus reveal its relationship with NTN. This mathematical explanation enables us to have a better viewpoint regarding NTN.

Taylor's theorem, named after mathematician Brook Taylor, is first proposed in 1712. It proves that a kth order Taylor polynomial (Fig. 1) approximates a kth order differentiable function around a given point. Based on Taylor's theorem, we analyze NTN from the perspective of function approximation. That is, each slice in NTN is considered to approximate an unknown function that captures a relation between two vectors. This is because there is a strong connection between NTN and Taylor polynomial which provides a feasible method to approximate that function without knowing the exact form. Moreover, it is also the theoretical basis for subsequent improvements on NTN. In summary, the scientific contributions of this paper are as follows:

1. We present the mathematical explanation for NTN from the perspective of Taylor's theorem and bring two different forms of neural tensor together into a single framework.
2. We reveal the inner link between NTN and other models, e.g., feedforward neural networks and attention mechanism, and factorize NTN for parameter-efficiency.
3. We conduct empirical studies on three NLP tasks to analyze the performance of NTN and obtain some important insights.

The remainder of the paper is organized as follows: Section 2 introduces related work; Section 3 presents the mathematical analysis of NTN; Section 4 discusses empirical studies and results; finally, Section 5 proposes concluding remarks.

2. Related work

NTN is an expressive neural network architecture and was first proposed by [10] for knowledge base completion task. NTN

* Corresponding author.
E-mail addresses: wei008@e.ntu.edu.sg (W. Li), luyao001@e.ntu.edu.sg (L. Zhu), cambria@ntu.edu.sg (E. Cambria).

https://doi.org/10.1016/j.knosys.2021.107258
0950-7051/© 2021 Elsevier B.V. All rights reserved.

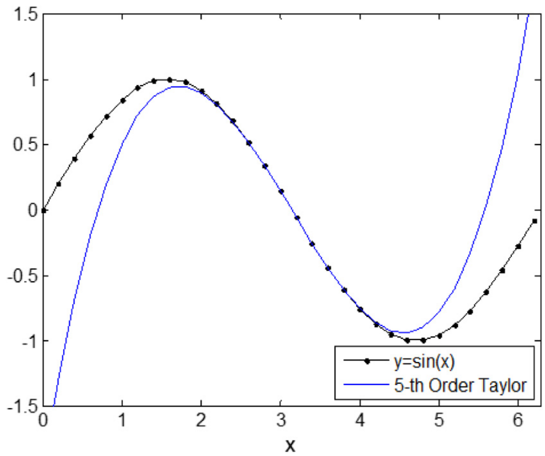

Fig. 1. The trigonometric function $y = sin(x)$ (black) and the corresponding Taylor polynomial of degree five (blue) around the point π.

is a generalization of several previous studies [11–14] on entity representation and relation modeling and has a good capability of modeling relational information. In some more complicated situations, a tensor is used to capture multi-modal relations [15]. In general, an entity, character, word, sentence or document is represented as a sparse or dense vector for computation in most NLP tasks. The relationship between two entities, words or sentences is modeled as the interaction between two vectors. In this case, NTN is applied to capture the relationship between two vectors. For instance, Socher et al. [10] first utilized NTN for reasoning over relationships between two entities on knowledge base completion task. Moreover, NTN is extended to associate a sequence of vectors by means of a recursive mechanism. For example, Socher et al. [7] proposed an NTN based recursive deep model to associate all the words vectors in a sentence or document to conduct semantic compositionality on sentiment analysis task. Recently,Li et al. [8] applied NTN to extract context information for a given utterance vector on conversational sentiment analysis task.

Although some previous studies report that NTN is more powerful in modeling relational information than the feedforward neural networks [10], the latter still has its unique strengths. For instance, feedforward neural networks have fewer parameters compared with NTN and are faster in the training phase. Therefore, a thorough theoretical analysis is needed to clarify the relationship between NTN and feedforward neural networks. [16] converted NTN to a multilayer perceptron (MLP) based representation, bringing some novel insights regarding NTN. They found that NTN can be viewed and represented as a two-layer feedforward neural network in its traditional form. However, the theoretical basis of such representation of NTN remains unclear. Our research reveals the inner link between NTN and other models, e.g., feedforward neural network and attention mechanism, from the perspective of Taylor's theorem.

3. Neural tensor network

NTN was proposed by [9,17] in knowledge base models to represent the relations between two entities. The proposed models outperform the previous models on knowledge representation and reasoning tasks. NTN was also employed in other studies, e.g., visual question answering [2], community-based question answering [3], and implicit discourse relation recognition [18]. The model is used to represent whether two entities (e_i, e_j) are in a certain relation R [17]. For example, NTNs are capable of stating whether relation (e_i, R, e_j) =(*Max, love, Cynthia*) is true and with what certainty, where $e_i, e_j \in \mathbb{R}^{d_e\times 1}$ are vectors of the two entities. The original NTN is shown as the following function:

$$h(e_i, R, e_j) = u^T f(e_i^T \underline{M_R^{[1:k]}} e_j + V_R \begin{bmatrix} e_i \\ e_j \end{bmatrix} + b_R), \tag{1}$$

where f is a standard nonlinear function applied element-wise, e.g., tanh, sigmoid, $\underline{M_R^{[1:k]}} \in \mathbb{R}^{d_e\times d_e\times k}$ is a tensor, d_e is the dimension of the entity. $V_R \in \mathbb{R}^{k\times 2d_e}$, $e_i, e_j \in \mathbb{R}^{d_e\times 1}$, $u^T, b_R \in \mathbb{R}^{k\times 1}$. $e_i^T M_R^{[m]} e_j$ is a computed one slice of the tensor layer $e_i^T \underline{M_R^{[1:k]}} e_j$, $m = 1, 2, \ldots, k$, which is considered as a "feature extractor" capturing the interactions between e_i and e_j. To be specific, $e_i^T M_R^{[m]} e_j$ captures a specific relationship between entity e_i and e_j.

As the wide application of NTN in a variety of tasks [5,19], another variation was introduced:

$$h(e, R) = u^T f(e^T \underline{M_R^{[1:k]}} e + V_R e + b_R), \tag{2}$$

where $e \in \mathbb{R}^{nd_e\times 1}$ is the concatenated feature embeddings vector, and $n \in \mathbb{N}^+$ is the number of entities, which usually equals 2 when $e = [e_i^T, e_j^T]^T \in \mathbb{R}^{2d_e\times 1}$, $\underline{M_R^{[1:k]}} \in \mathbb{R}^{2d_e\times 2d_e\times k}$. Besides, the use of u^T varies in different research tasks. For example, u^T is applied in NTN function to generate a scalar for two-class classification of relations in knowledge base completion task shown above. While in other works, e.g., question answering task, $f(e^T \underline{M_R^{[1:k]}} e + V_R e + b_R)$ is regarded as a NTN architecture, i.e., u_T is omitted; Because in this case, the model is tailored for feature extraction for semantic compositionality [7].

$$h(e, R) = f(e^T \underline{M_R^{[1:k]}} e + V_R e + b_R) \tag{3}$$

In the following part of this paper, the term NTN refers to formula (3), which is also the key research object in this work.

3.1. Taylor neural network slice

In this subsection, the architecture of NTN will be illustrated through detailed examples firstly. Then, we show that NTN is equivalent to Taylor polynomial under certain conditions, which provides a new perspective for the explanation of NTN. Based on this consideration, we propose a Taylor Neural Network Slice (TNNS) framework to provide guidance for the construction and application of NTN.

In addition to the tasks described above, NTN in formula (3) is applied to extracting context features for sentiment analysis in dialogues [8]. The same example in the introduction section is illustrated as follows.

Given two adjacent utterances in a dialogue, u_1-"The film is terrible"., u_2-"Yes, except for the beginning"., a model is designed to classify the sentiment polarity of the second utterance. However, it is difficult to predict u_2's polarity without the comprehension of u_1. Thus, NTN in this paper is used to extract the interaction between u_1 and u_2, and the context information C of u_2.

First the two utterances u_1, u_2 are represented as two vectors $e_1, e_2 \in \mathbb{R}^{d_e\times 1}$ through a specific model. $e^T = [e_1^T, e_2^T] \in \mathbb{R}^{1\times 2d_e}$ is the concatenated vectors of the two utterances. In this example, NTN is illustrated as formula (4).

$$\begin{aligned} h(e, C) &= f(b_C + V_C e + e^T \underline{M_C^{[1:k]}} e) \\ &= f(b_C + V_C \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + [e_1^T, e_2^T] \underline{M_C^{[1:k]}} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}) \end{aligned} \tag{4}$$

Suppose NTN is composed of 3-slices tensor layer and the utterance embedding is 2-dimensional, i.e. $k = 3, d_e = 2$, then NTN could be represented as:

$$h(e, C) = f(t), \quad t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{5}$$

$$
\begin{aligned}
t_i &= b_{iC} + V_{iC}\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + [e_1^T, e_2^T] M_C^{[i]} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \\
&= b_i + \begin{bmatrix} v_{i1} & v_{i2} & v_{i3} & v_{i4} \end{bmatrix} \begin{bmatrix} e_{11} \\ e_{12} \\ e_{21} \\ e_{22} \end{bmatrix} \\
&+ \begin{bmatrix} e_{11} & e_{12} & e_{21} & e_{22} \end{bmatrix} \begin{bmatrix} m_{11}{}^i & m_{12}{}^i & m_{13}{}^i & m_{14}{}^i \\ m_{21}{}^i & m_{22}{}^i & m_{23}{}^i & m_{24}{}^i \\ m_{31}{}^i & m_{32}{}^i & m_{33}{}^i & m_{34}{}^i \\ m_{41}{}^i & m_{42}{}^i & m_{43}{}^i & m_{44}{}^i \end{bmatrix} \begin{bmatrix} e_{11} \\ e_{12} \\ e_{21} \\ e_{22} \end{bmatrix}, \\
& i = 1, 2, 3.
\end{aligned} \tag{6}
$$

$M_C^{[i]}$ corresponds to the ith slice of tensor $M_C{}^{[1:3]}$. The detailed architecture of NTN in formula (6) is shown in Fig. 2, where Fig. 2(a) shows the calculation process of t_i in formula (6) and Fig. 2(b) corresponds to formula (5).

According to Fig. 2 and formula (6), it is interesting to find that the representation of t_i is the 2nd order Taylor polynomial for functions of multiple variables. Thus, the t_i in formula (6) is also named as the 2nd order Taylor Neural Network Slice (TNNS) of NTN in this paper. When $M_C^{[i]} = O$ (zero matrix), t_i refers to the 1st order Taylor Neural Network Slice, which equals the 1st order Taylor polynomial; and in this case, $f(t_i)$ is a 1st order NTN and a feedforward neural network as well. According to Taylor's theorem, for a multivariate function $g(x)$, of which the value is a scalar, $x \in \mathbb{R}^{d\times 1}$, the first derivative is $\nabla g(x) \in \mathbb{R}^{d\times 1}$, and the second derivative is the Hessian matrix of $g(x)$, which is denoted as $Hf(x) \in \mathbb{R}^{d\times d}$. Here, $d \in \mathbb{N}^+$ describes the dimension of the multiple variable x. Thus, the 2nd order Taylor polynomial for multivariate function $g(x)$ around the point $x^{(0)}$ is:

$$
\begin{aligned}
g(x) &= g(x^{(0)}) + \nabla g(x^{(0)})(x - x^{(0)}) \\
&+ \frac{1}{2}(x - x^{(0)})^T Hg(x^{(0)})(x - x^{(0)}) + o((x - x_0)^3),
\end{aligned} \tag{7}
$$

where $\nabla g(x^{(0)}) = [\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}, \dots, \frac{\partial g}{\partial x_d}]\Big|_{x^{(0)}}^T$,

$$
Hg(x^{(0)}) = \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_1 \partial x_d} \\ \frac{\partial^2 g}{\partial x_2 \partial x_1} & \frac{\partial^2 g}{\partial x_2^2} & \cdots & \frac{\partial^2 g}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 g}{\partial x_d \partial x_1} & \frac{\partial^2 g}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_d^2} \end{bmatrix}.
$$

For $d = 2$, $x = \left[x_1, x_2\right]^T$, $x^{(0)} = \left[x_1^{(0)}, x_2^{(0)}\right]^T$, $\Delta x = x - x^{(0)} = \left[\Delta x_1, \Delta x_2\right]^T = \left[x_1 - x_1^{(0)}, x_2 - x_2^{(0)}\right]^T$ then the formula (7) could be written as:

$$
\begin{aligned}
g(x) &= g(x^{(0)}) + \begin{bmatrix} \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} \end{bmatrix}_{x^{(0)}} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} \\
&+ \frac{1}{2}\begin{bmatrix} \Delta x_1 & \Delta x_2 \end{bmatrix} \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_1 \partial x_2} \\ \frac{\partial^2 g}{\partial x_2 \partial x_1} & \frac{\partial^2 g}{\partial x_2^2} \end{bmatrix}_{x^{(0)}} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} + o(\Delta x^3)
\end{aligned} \tag{8}
$$

As shown, the t_i block, namely the 2nd order TNNS, in formula (6) accounts for approximating the 2nd order Taylor polynomial for a certain multivariate function $g(x)$ at the point $x^{(0)}$, with b_C, V_C, $M_C^{[i]}$, and e corresponding to $g(x^{(0)})$, $\nabla g(x^{(0)})$, $\frac{1}{2}Hg(x^{(0)})$, and Δx, respectively. This may provide the explanation why NTN works.

Furthermore, a 3rd order TNNS is proposed in this work for context information extraction under the guideline of Taylor's theorem.

$$
t_i = b_C + V_C e + e^T M_C^{[i]} e + \underline{P_C^{[i]}} \overline{\times}_1 e \overline{\times}_2 e \overline{\times}_3 e \tag{9}
$$

The $\underline{P_C^{[i]}} \in \mathbb{R}^{2d_e\times 2d_e\times 2d_e}$ is the ith slice, more precisely, subtensor of the 4th order tensor $\underline{P_C^{[1:k]}} \in \mathbb{R}^{2d_e\times 2d_e\times 2d_e\times k}$. According to [20], $\underline{C} = \underline{A} \overline{\times}_j b$ is a mode-j product of Nth order tensor $\underline{A} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ and vector $b \in \mathbb{R}^{I_j}$, which yields a tensor $\underline{C} \in \mathbb{R}^{I_1\times\cdots\times I_{j-1}\times I_{j+1}\times\cdots\times I_N}$, with entries $c_{i_1,\dots,i_{j-1},i_{j+1},\dots,i_N} = \sum_{i_j=1}^{I_j} a_{i_1,\dots,i_j,\dots,i_N} b_{i_j}$. Thus, $\underline{P_C^{[i]}} \overline{\times}_1 e \overline{\times}_2 e \overline{\times}_3 e = \sum_{i=1}^{2d_e}\sum_{j=1}^{2d_e}\sum_{k=1}^{2d_e} p_{ijk} e_i e_j e_k$. Obviously, tensor $\underline{P_C^{[i]}}$ is devised to fit the 3rd derivative of function $g(x)$ at the point $x^{(0)}$, i.e., p_{ijk} approximates $\frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k}$.

3.2. Factorization for neural tensor networks

Although NTN shows good performance on different tasks, the model faces the curse of dimension as the increase of the dimension d_e of entity e. It is obvious that the reduction of the computation complexity of the tensor $M_C^{[1:k]}$ would relieve the model of this problem. Besides, since the tensor $M_C^{[1:k]}$ is composed of k slices of matrix $M_C^{[i]}$, the key step is to reduce the computation complexity of $M_C^{[i]}$. As stated in Section 3.1, $M_C^{[i]}$ corresponds to $\frac{1}{2}Hg(x^{(0)})$ in Taylor's theorem. Thus, the factorization of the tensor slice $M_C^{[i]}$ is based on the discussion of the $Hg(x^{(0)})$ case by case.

Case One. ($M_C^{[i]} \neq M_C^{[i]^T}$) If $M_C^{[i]}$ is a real asymmetric matrix, it could be decomposed via Singular Value Decomposition (SVD) [21]. i.e. $\exists\ U, V^T \in \mathbb{R}^{[2d_e\times 2d_e]}$, satisfying $M_C^{[i]} = U\Sigma V^T$, where U, V are unitary matrices, the diagonal elements of Σ are the singular values of $M_C^{[i]}$. Based on previous work and applications of SVD [22], $M_C^{[i]}$ could be approximated as $M_C^{[i]} = U_{(k)}\Sigma_{(k)}V_{(k)}^T$, where $\Sigma_{(k)}$ is the k-order principal minor of Σ, and $U_{(k)}, V_{(k)}$ correspond to the first k columns of U, V, respectively, $1 \le k \le 2d_e$.

Case Two. ($M_C^{[i]} = M_C^{[i]^T}$) If $M_C^{[i]}$ is a real symmetric matrix, it could be diagonalized, i.e. $\exists\ P \in \mathbb{R}^{[2d_e\times 2d_e]}$, $P^T = P^{-1}$, satisfying $M_C^{[i]} = P^T \Lambda P$, where P is also named as unitary matrix. Based on Case One, $M_C^{[i]}$ could be approximated as $M_C^{[i]} = P_{(k)}\Lambda_{(k)}P_{(k)}^T$, where $\Lambda_{(k)}$ corresponds to the k-order principal minor of Λ, and $P_{(k)}$ is the first k columns of P.

A sufficient condition of Case Two is that the multivariate function $g(x)$ is of class C^k, i.e., all the i-order partial derivatives exist and are continuous, for $\forall i \le k$, $i, k \in \mathbb{N}^+$.

3.3. Relationship between two forms of neural tensor networks

As shown in the aforementioned introduction of formulas (1) and (2), both formulas are referred to as NTN in different research papers. Besides, both of them could be applied to relation feature extraction. Moreover, we found an interesting relationship between NTN in formula (1) and that in formula (2): formula (1) is a special case of formula (2) under certain conditions.

As known, the only difference between formula (1) and formula (2) is that between tensor layer slices $e_i^T M_R^{[m]} e_j$ and $e^T M_R^{[m]} e$, where $e = [e_i^T, e_j^T]^T \in \mathbb{R}^{2d_e\times 1}$. Thus, we focus on exploring the relationship between the two tensor layer slices. For brevity, we use $W = \begin{bmatrix} W_1 & W_2 \\ W_3 & W_4 \end{bmatrix} \in \mathbb{R}^{2d_e\times 2d_e}$ to represent $M_R^{[m]}$, where $W_l \in \mathbb{R}^{d_e\times d_e}$, $l = 1, 2, 3, 4$.

$$
\begin{aligned}
e^T W e &= [e_i^T, e_j^T] W \begin{bmatrix} e_i \\ e_j \end{bmatrix} \\
&= [e_i^T, e_j^T] \begin{bmatrix} W_1 & W_2 \\ W_3 & W_4 \end{bmatrix} \begin{bmatrix} e_i \\ e_j \end{bmatrix} \\
&= e_i^T W_1 e_i + e_j^T W_3 e_i + e_i^T W_2 e_j + e_j^T W_4 e_j \\
&= e_i^T (W_2 + W_3^T) e_j + e_i^T W_1 e_i + e_j^T W_4 e_j
\end{aligned} \tag{10}
$$

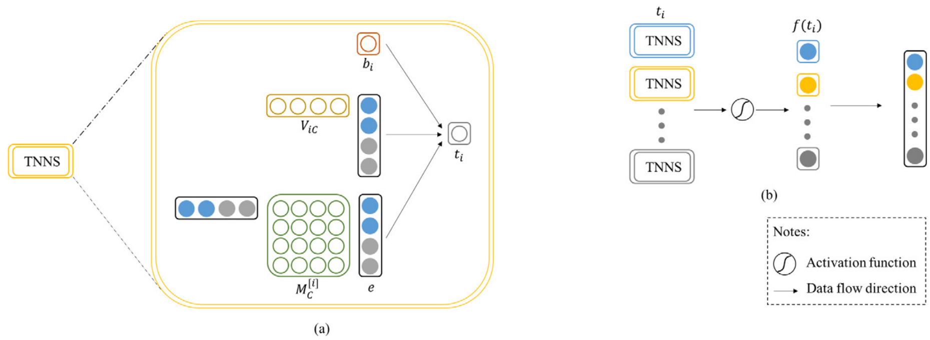

Fig. 2. The representation of NTN with two entities initialization. (a) The architecture of the 2nd order Taylor Neural Network Slice (TNNS). (b) The representation of NTN with k slices of TNNS.

According to formula (10), if $W_1 = W_4 = O$ (zero matrix), we found that

$$e^T We = e_i^T (W_2 + W_3^T) e_j. \tag{11}$$

This is the same as the form of tensor layer slice $e_i^T M_R^{[m]} e_j$, which proves that the formula (1) is a special case of formula (2). Therefore, it is sensible to focus on the representation in formulas (2) and (3) in this paper when referring to NTN.

3.4. Relationship between neural tensor network and attention mechanism

In recent years, the attention mechanism has been widely utilized to improve the performance of deep learning models on various NLP tasks, such as neural machine translation [23,24], document classification [25], album summarization [26], etc. According to [27], the common choices [23] for computing the attention score s_i in basic attention are given by:

Dot:

$$s_i = q_t^T k_i, \tag{12}$$

General:

$$s_i = q_t^T W_g k_i, \tag{13}$$

Additive (Multi-layer perceptron):

$$\begin{aligned} s_i &= u^T tanh(W_q q_t + W_k k_i + b) \\ &= u^T tanh(W_a [q_t; k_i] + b), \end{aligned} \tag{14}$$

where $q_t \in \mathbb{R}^{d_q}$ is a query vector (or target hidden state), $k_i \in \mathbb{R}^{d_k}$ is the ith key vector (or source hidden state); in most cases, we set $d_q = d_k = d$; then, weight matrices W_g, W_q, $W_k \in \mathbb{R}^{d\times d}$, $W_a \in \mathbb{R}^{m\times 2d}$, bias vector $b \in \mathbb{R}^m$, and $u \in \mathbb{R}^m$.

We found that the general attention $q_t^T W k_i$ is a special case of the 2nd order TNNS term, and the dot attention is a special case of the general attention $q_t^T W_g k_i$, when $W_g = I$; besides, the additive attention $u^T tanh(W_a[q_t; k_i] + b)$ is the 1st order NTN. Moreover, [28] proposed the self-attention mechanism, becoming a research focus in these years. The attention score in the self-attention mechanism is:

$$s_{ij} = \frac{(x_i W_Q)(x_j W_K)^T}{\sqrt{d_k}}. \tag{15}$$

Here, x_i, $x_j \in \mathbb{R}^{d_x}$, and W_Q, $W_K \in \mathbb{R}^{d_x\times d_k}$. It is obvious that the self-attention score utilizes the 2nd order TNNS term with SVD decomposition (Case One in Section 3.2).

According to the formulas above, there is a strong connection between NTN and the attention score function, since both models employ TNNS which enables them to approximate any nonlinear functions without knowing the exact forms.

4. Experiments

In this section, we conduct experiments on several NLP tasks including conversational sentiment analysis (CSA), named entity recognition (NER) and knowledge base completion (KBC) to testify the aforementioned mathematical analysis of NTN.

4.1. Task definition and parameter setting

Conversational sentiment analysis. The goal of CSA [29–31] is to predict the sentiment polarities (e.g., frustrated, neutral, sad and happy) of each utterance in a conversation. However, context utterances may sometimes enhance, weaken or reverse the raw sentiment of an utterance. Therefore, we use NTN to associate an utterance with its context utterances to incorporate the context information into the current utterance. Then the current utterance is fed into a long short-term memory (LSTM) block followed by a fully connected layer for classification. We also report the performance of the following baselines for comparison: c-LSTM [32] and DialogueRNN [33]. Here, we follow the experiment protocols as described in [33], and use identical feature extraction procedures converting utterances into vectors. We set the dimension of the utterance vector as 50, the number of slices as 50, batch size as 1. We empirically set the learning rate as 0.0001, the input dimension of NTN as 100. L2 regularization and dropout [34] are employed to alleviate over-fitting. We use the factorization method case one mentioned in Section 3.2 to reduce trainable parameters. The neural network is optimized by an Adam Optimizer [35].

Named entity recognition. NER is an important information extraction task that requires identifying and classifying pre-defined named entity categories such as *location*, *organization* and *person* in a given text [36,37]. In this paper, we employ the classical end-to-end sequence labeling model bidirectional LSTM-CNNs-CRF [38] as the base model to perform the NER task on the standard CoNLL NER dataset [39]. This NER system takes use of concatenated word-level embedding and character-level representation as the input feature, where the character representation is computed by a convolutional neural network (CNN) [40]. Then, the input feature is fed into a bidirectional long short-term memory (BiLSTM) [41] network and the output of BiLSTM is fed to a conditional random field (CRF) [42] layer to jointly decode the optimal label sequence. In the experiment, we utilize NTN instead of concatenation operation to better combine word embedding and character representation. Similarly, we follow the experiment protocols as described in [38], and use identical CNN for character-level representation. The dimension is 100 and 50 for word embedding and character-level representation respectively. The dimension of word embedding and character-level representation are set to 100 and 50 respectively. The number of slices is set to 150. We empirically set learning rate as 0.015 and decay rate as 0.05. Similarly, factorization method case one mentioned in Section 3.2 is utilized to improve the computation-efficiency of NTN. The model is optimized by a stochastic gradient descent (SGD) optimizer with a momentum of 0.9.

Knowledge base completion. Knowledge bases like WordNet [43], SenticNet [44] or Google Knowledge Graph are of vital importance for query expansion [45], question answering (Google assistant) or giving structured knowledge to users. However, knowledge bases often suffer from incompleteness, generating the need for KBC [46]. Most studies focus on extending the existing knowledge base employing patterns or classifiers applied to large corpora [47,48]. However, complex or rare knowledge is not as usual as the common knowledge in text. Therefore, commonsense reasoning which refers to predicting the likely truth of additional facts based on existing facts in the knowledge base [49], is useful and available for users to obtain rare or complex knowledge in text. We take the classical model [10] as the base model to perform relation triplet classification. In this model, a neural tensor layer described in formula (1) is used to explicitly relate two entities. Furthermore, we extend this special case to a general case as described in formula (2) for comparison. We use Turian [50] initialization to initialize entity and relation embedding, and set the dimension of their embedding as 100. We set the number of slices as 3, batch size as 20000, corrupt size as 10. We train the model for 500 iterations and use L-BFGS [51] as the optimizer in this experiment.

In general, there is no one-size-fit-all hyper-parameter setting that can cope with different tasks. In most cases, we need to fine-tune the hyper-parameters of NTN when applying it to different tasks. Parameter analysis and grid search are two good methods used to obtain the optimal hyper-parameters of NTNs.

4.2. Datasets

In this subsection, we introduce all the datasets used in the three NLP tasks among which IEMOCAP [52] and MELD [53] are for CSA, CoNLL2003 [39] for NER, and WordNet [43] and Freebase [54] for KBC. To be specific, IEMOCAP and MELD are the most commonly used benchmark datasets for CSA datasets; CoNLL2003 is one of the most famous NER datasets released at the top tier conference CoNLL; Similarly, WordNet and Freebase are often used for knowledge base completion task since the release.

IEMOCAP. IEMOCAP is a dataset composed of two-way conversations with ten distinct participators. Each utterance in a conversation is marked by one of the six sentiment labels, namely *happy*, *sad*, *neutral*, *angry*, *excited* and *frustrated*. In the experiments, we only focus on textual modality data (details in Table 1).

MELD. MELD consists of textual, acoustic, and visual information from more than 13000 utterances from Friends TV series. It is a multiparty and multimodal sentiment classification dataset. The sentiment label of each utterance comes from one of the following seven labels: *joy*, *surprise*, *sadness*, *fear*, *neutral*, *anger* and *disgust* (details in Table 1).

CoNLL. The standard CoNLL dataset contains four types of name entities. i.e., *person*, *location*, *organization* and *misc*. Here we use BIOES tagging schema instead of the BIO2 since previous studies reported significant improvement with this schema [55]. The statistical information of this dataset is shown in Table 2.

Table 1
Statistical information of IEMOCAP and MELD datasets.

Dataset	Partition	Dialogue count	Utterance count
IEMOCAP	Train + val	120	5,810
	Test	31	1,623
MELD	Train + val	1,153	11,098
	Test	280	2,610

Table 2
Statistical information of CoNLL2003 dataset.

Dataset	Sentence count	Token count
Train	14,987	204,567
Dev	3,466	51,578
Test	3,684	46,666

Table 3
The statistics for WordNet and Freebase.

Dataset	Partition	Count	Relation
WordNet	Train	122,581	11
	Dev	2,609	11
	Test	10,544	11
Freebase	Train	316,232	13
	Dev	5,908	7
	Test	23,733	7

WordNet. WordNet database is composed of 112,581 relational triplets for training. A triplet (e_1, *similar to*, e_2) comprises two entities (e_1, e_2) and a relation *similar to* between them. In the experiment, we filter out trivial test triplets and obtain 38,696 unique entities in 11 different relations. Besides, the triplets appeared in both training and testing sets in a different relation or order are filtered out. For example, (e_2, *similar to*, e_1) and (e_1, *type of*, e_2) are removed if (e_1, *similar to*, e_2) is in the training set. We display statistical information of WordNet in Table 3.

Freebase. We use the relational triplets from *People* domain and obtain 316,232 triplets in 13 relations for training. However, among these 13 relations, *place of death*, *place of birth*, *location*, *parents*, *children* and *spouse* are quite hard to predict and are removed from the testing set. Table 3 shows the statistics of Freebase.

4.3. Neural tensor network variants

Following the theoretical guide of Taylor's theorem, we propose several variants of NTN.

The first order neural tensor networks. This variant uses only the zero-order term and the 1st order term of TNNS. The computation formula is $f(b_R + V_R e)$.

The second order neural tensor networks. The 2nd order NTN is the same as traditional NTNs, where the zero, 1st and 2nd order terms of TNNS are included for computation. The computation formula is $f(b_R + V_R e + e^T \underline{M_R^{[1:k]}} e)$.

The third order neural tensor networks. We also introduce the 3rd order term based on Taylor's theorem. The computation formula of this variant is shown in Eq. (9).

4.4. Experimental results

We report the empirical results of NTN variants as well as baseline methods on IEMOCAP & MELD datasets of CSA task, CoNLL dataset of NER task and WordNet & Freebase datasets of KBC task, respectively. The experimental results are shown in Tables 4, 5, 6 and 7.

Table 4
Experiment results on IEMOCAP and MELD datasets. Bold font denotes the best performance. Acc. = Accuracy; Average(w) = Weighted average.

Methods	IEMOCAP														MELD
	Happy		Sad		Neutral		Angry		Excited		Frustrated		Average(w)		Acc.
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	
c-LSTM	30.6	**35.6**	56.7	62.9	57.6	53.0	59.4	59.2	52.8	58.9	**65.9**	59.4	56.3	56.2	57.5
DialogueRNN	25.7	33.2	75.1	78.8	58.6	59.2	64.7	65.3	**80.3**	71.9	61.2	58.9	63.4	62.8	56.7
The 1st order NTN	45.8	30.6	80.0	83.0	67.8	59.9	67.9	65.7	61.6	72.7	59.4	60.4	65.1	63.8	59.9
The 2nd order NTN	**46.3**	29.4	77.5	82.7	**69.0**	58.4	**68.1**	**66.1**	61.7	**72.9**	58.9	60.6	65.0	63.5	59.9
The 3rd order NTN	44.2	30.8	**83.3**	**84.5**	66.4	**60.7**	67.5	65.5	61.6	71.9	60.6	**61.4**	**65.4**	**64.4**	**60.2**

IEMOCAP result. As evidenced by Table 4, for the IEMOCAP dataset, NTN variants outperform baselines DialogueRNN and c-LSTM by a large margin on average. The performance of the 1st order NTN exceeds DialogueRNN by 1.7% in accuracy and 1.0% in f1-score on average. The major constituent of the 1st order NTN is the 1st order TNNS approximating the 1st order differentiable functions, which accounts for the good performance of the 1st order NTN. Besides, the relationships between the context utterance and the current utterance are not as complex as the entity relationships in some other NLP tasks so that the 1st order NTN is good enough for CSA. The 2nd order NTN gets almost the same average accuracy and f1-score as the 1st order NTN. This means the introduction of the 2nd order term of TNNS did not significantly improve the overall performance. As for the 3rd order NTN, its performance surpasses the 2nd order NTN by 0.4% in accuracy and 0.9% in f1-score on average. We think that the reason for such improvement is that the introduction of the 3rd order term of TNNS enables NTN to approximate more complicated functions. However, the high order terms dramatically increase the trainable parameters with limited performance gain, which is not computation-efficient in this task. The 2nd order NTN without factorization gets 64.76% in accuracy and 63.54% in f1-score on the IEMOCAP dataset respectively, which is almost the same as the one with factorization. On the MELD dataset, the 2nd order NTN without factorization achieves a slightly higher accuracy (60.2%) than that (59.9%) with factorization. In conclusion, factorization methods significantly reduce trainable parameters of NTN without decreasing the overall performance.

MELD result. NTN variants surpass baseline methods by a large margin on the multiparty conversation dataset MELD. However, there is no big difference in the performance of NTN variants. The introduction of the 2nd order terms of TNNS does not increase the classification accuracy. The 3rd order NTN slightly improves the classification accuracy, due to the 3rd order term of TNNS. Similarly, the performance of the 1st order NTN is good enough. One possible reason is that the relationships between the context utterance and the current utterance are not very complicated so that the 1st order NTN can capture most of these relationships. Another reason is that the average MELD conversation length is 10 utterances, with more than 5 participants in many conversations, which means that the connections between the current utterance and the adjacent utterances are further weakened. In this case, it does not make much sense to use the high order NTN to approximate complicated functions.

Ablation study and parameter analysis. To further study the influence of each term of NTN on the CSA task, we perform an ablation study on the IEMOCAP dataset and display the results in Table 5. It indicates that the 1st order term plays an important role in context compositionality of the CSA task. If removing the 1st order term from NTN, its performance on the IEMOCAP dataset drop dramatically. The 2nd and 3rd order terms may slightly improve or weaken the performance of NTN. There are two possible reasons for this phenomenon. Firstly, the 2nd and 3rd order terms introduce a large number of parameters that require more data samples in the training phase. Secondly, the number of slices is enough for the 1st order NTN to capture the interactions between two utterances so that the gain of the 2nd and 3rd order terms are relatively small. We also conduct parameter analysis for the number of slices on the IEMOCAP dataset. As shown in Fig. 3, the influence of slice number on three NTN variants are almost the same. The performance of NTN variants improves dramatically as slice number increases from 5 to 15, which indicates that employing more slices enables the model to capture more relationships between two utterances from different perspectives. Besides, the accuracy of the 3rd order NTN is slightly better than those of the 2nd order and 1st order NTN within the interval [5, 15]. Then, the performance of NTN variants fluctuates from 63.5% to 65.5%, which shows that slice number is enough and adding additional slice cannot further improve the overall performance to some extent. Here, the optimal interval of slice number ranges from 45 to 55 based on the parameter analysis. Nevertheless, the optimal parameter interval of slice number may vary from dataset to dataset as well as task to task.

Table 5
Results of ablated NTN on IEMOCAP dataset. Accuracy and F1-score are weighted average.

The 1st order term	The 2nd order term	The 3rd order term	Acc.	F1-score
+	−	−	65.13	63.84
+	+	−	65.00	63.47
+	+	+	65.43	64.37
−	+	−	24.68	14.62
−	+	+	23.66	9.05
−	−	+	24.58	11.53

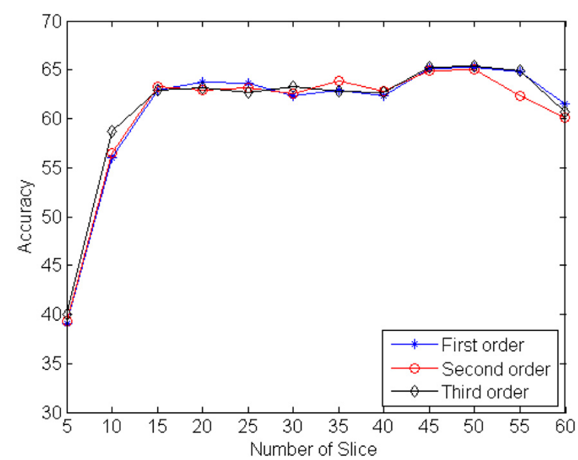

Fig. 3. Accuracy of three NTN variants on IEMOCAP dataset with different number of slices.

CoNLL result. For NER, we employ NTN to combine the pre-trained 100 dimensional word-level embedding e^{w}[1] and CNN or

[1] https://nlp.stanford.edu/projects/glove/.

Table 6
Experimental results on CoNLL dataset.

Method	Character Embedding	Dev F1-score	Test F1-score
The 1st order NTN	BiLSTM	0.875	0.798
	CNN	0.869	0.795
The 2nd order NTN	BiLSTM	**0.883**	**0.810**
	CNN	**0.879**	**0.809**

Table 7
Relation triplet classification results on WordNet and Freebase datasets.

Method	WordNet	Freebase
The 1st order NTN	0.704	0.824
The 1st order NTN*	0.695	0.818
The 2nd order NTN (form two)	0.831	0.837
The 2nd order NTN* (form two)	0.824	0.826
The 2nd order NTN (form one)	0.827	0.828
The 2nd order NTN* (form one)	0.817	0.832
The 2nd order term (form two)	0.837	0.663
The 2nd order term* (form two)	0.840	0.737
The 2nd order term (form one)	0.848	0.672
The 2nd order term* (form one)	0.810	0.711

*is a mark for identity activation function.

BiLSTM generated 50 dimensional character-level representation e^c.[2] In general, character-level representation is less informative and is considered complementary for word-level embedding. As shown in Table 6, the overall performance of the 2nd order NTN is slightly better than that of the 1st order NTN on both dev and test sets. The structure of the LSTM-CNNs-CRF model is quite complex and NTN accounts for only one layer of the whole model. Therefore, NTN variants have a small impact on the final classification results. Different from the CSA task, word and character embedding dimensions are not equal and $e^T = [e_i^{wT}, e_j^{cT}]$ in formula (2) are unbalanced in this case. This is an extension of the existing NTN models and complies with the theoretical basis of Taylor's theorem.

WordNet result. Both the aforementioned models for CSA and NER tasks have complicated structures where NTN accounts for only one layer of the whole model. In this case, the difference between the performance of NTN variants is relatively minor. To this end, we conduct experiments on KBC to directly study the properties of NTN variants. To be specific, our goal is to predict correct facts in the manner of relations (e_1, R, e_2) in the testing dataset. As illustrated in Table 7, the introduction of the 2nd order TNNS term dramatically improves the triplet classification results on the WordNet dataset. We also compare the performance of two different forms of the 2nd order NTN, where form one is described in formula (1) and form two is defined in formula (2). The result indicates that form two has slightly better accuracy as compared with form one. One possible reason is that the former has more trainable parameters. Supposing that the dimension of the form one is (d, d, k), where d is the dimension of entity embedding and k the slice number. In this case, the dimension of form two is $(2d, 2d, k)$. Given $d = 100$, $k = 3$ and $Relation = 11$, the parameters of form one and form two are 330 K and 1320 K respectively. However, it is more sensible to use form one instead of form two in this situation considering the limited improvement in accuracy and rapidly increasing parameters. The default activation function is *tanh* and we replace it with identity activation function. Experimental results show that the influence of activation function on the performance of NTN variants is relatively small in most cases. For the 1st order NTN, *tanh* activation function improves the overall performance by a small margin. This is because the activation function increases the representation capability for approximating non-linear functions. However, the influence of the activation function on the 2nd order NTN is more complex. The 2nd order TNNS term provides non-linear representation ability for the models. Therefore, some 2nd order models with identity activation even get better performance on both datasets.

Freebase result. As shown in Table 7, form one and form two get almost the same results on the Freebase dataset, which shows that the 2nd order NTN models are competent for relation triplet classification. It is worth noting that the numbers of relations on training and testing sets are different and six relations that are hard to predict are removed from the testing set according to [10]. In this case, the 2nd order NTN models, namely form one and form two, outperform the 1st order NTN by only a small margin. Nevertheless, given the optimal hyper-parameters, the accuracy of form one surpasses the 1st order NTN by about 5% (reported by [10]). Besides, if we only use the 2nd order NTN term for classification, then form one and form two become special cases of the 2nd order NTN models where V_R and b_R are k slices of zero vectors. However, the performances of such special cases are not stable across different datasets. They are slightly better than the 2nd order NTN models on WordNet while are largely worse than the latter on Freebase, which indicates that the performances of special cases strongly depend on data. In contrast, the 2nd order NTN models show robust performance on both datasets, which demonstrates the superiority of the 2nd order NTN on the KBC task. Here, *tanh* and identity activation functions show a similar impact as they do on the WordNet dataset.

5. Conclusion

In this paper, we provide the mathematical explanation of NTN and also reveal the inner link between NTN and other models, e.g., feedforward neural network and attention mechanism, from the perspective of Taylor's theorem. In this situation, each Taylor neural network slice (TNNS) in NTN is regarded as a 2nd order multivariate Taylor polynomial while a 1st order NTN is equal to a feedforward neural network. Based on the theoretical analysis of NTN, we further propose factorization methods for parameter-efficiency. Moreover, we bring two forms of NTN together into a unified framework on the basis of the block matrix. Additionally, the 3rd order NTN is presented and tested in this paper.

The empirical studies performed on conversational sentiment analysis, named entity recognition and knowledge base completion tasks provide some valuable insights into NTN. Specifically, the 1st order NTN achieves competitive performance in most cases; the kth order TNNS introduces non-linearity into the kth order NTN when $k \geq 2$, which accounts for the good results of NTN with identity activation function; besides, form one obtains comparable performance against that of form two with much fewer parameters in most cases; the performance of the 2nd order NTN term is not stable across different datasets; when slice number is small, increasing the order of NTN benefits the performance whereas it is trivial to increase the order of NTN under the condition of a large slice number.

In summary, the mathematical deduction and the architecture analysis may enable readers to understand NTN more deeply and provide some guidance for designing NTN related models. Moreover, factorization is an effective attempt towards reducing the computation complexity of NTN. Last but not least, the proposed 3rd order TNNS term provides insights for the performance improvement of NTN related models.

[2] 100 dimensional word-level embedding and 50 dimensional character-level embedding are the default setting in [38].

Besides the aforementioned three NLP tasks, NTN related structures can also be used for some other tasks such as semantic compositionality, ranking the recommendations, community-based question answering. In general, NTN can be applied to any task involving the capture of entity relationships. In the future research, we plan to further discuss the optimal hyper-parameters in NTN based structures, and explore the relationships between NTN and some other deep learning structures.

CRediT authorship contribution statement

Wei Li: Software, Conceptualization, Methodology, Investigation, Validation, Writing - original draft, Writing - review & editing. **Luyao Zhu:** Methodology, Investigation, Writing - original draft, Formal analysis, Writing - review & editing. **Erik Cambria:** Supervision, Writing - review & editing, Resources, Funding acquisition, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research is supported by the Agency for Science, Technology and Research (A*STAR), Singapore under its AME Programmatic Funding Scheme (Project #A18A2b0046).

References

[1] Lirong He, Bin Liu, Guangxi Li, Yongpan Sheng, Yafang Wang, Zenglin Xu, Knowledge base completion by variational bayesian neural tensor decomposition, Cogn. Comput. 10 (6) (2018) 1075–1084.
[2] Yalong Bai, Jianlong Fu, Tiejun Zhao, Tao Mei, Deep attention neural tensor network for visual question answering, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 20–35.
[3] Xipeng Qiu, Xuanjing Huang, Convolutional neural tensor network architecture for community-based question answering, in: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, AAAI Press, 2015, pp. 1305–1311.
[4] Samuel R. Bowman, Can recursive neural tensor networks learn logical reasoning? in: 2nd International Conference on Learning Representations, ICLR 2014, Conference date: 14-04-2014 Through 16-04-2014, 2014.
[5] Wenzhe Pei, Tao Ge, Baobao Chang, Max-margin tensor neural network for Chinese word segmentation, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, pp. 293–303.
[6] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, Xiaolong Wang, Modeling mention, context and entity with neural networks for entity disambiguation, in: IJCAI, 2015, pp. 1333–1339.
[7] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, Christopher Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642.
[8] Wei Li, Wei Shao, Shaoxiong Ji, Erik Cambria, BiERU: Bidirectional emotional recurrent unit for conversational sentiment analysis, 2020, arxiv preprint arXiv:2006.00492.
[9] Danqi Chen, Richard Socher, Christopher D. Manning, Andrew Y. Ng, Learning new facts from knowledge bases with neural tensor networks and semantic word vectors, in: 1st International Conference on Learning Representations, ICLR 2013 ; Conference date: 02-05-2013 Through 04-05-2013, 2013.
[10] Richard Socher, Danqi Chen, Christopher D. Manning, Andrew Ng, Reasoning with neural tensor networks for knowledge base completion, in: Advances in Neural Information Processing Systems, 2013, pp. 926–934.
[11] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, Learning structured embeddings of knowledge bases, in: Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.
[12] Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, Guillaume R. Obozinski, A latent factor model for highly multi-relational data, in: Advances in Neural Information Processing Systems, 2012, pp. 3167–3175.
[13] Ilya Sutskever, Joshua B. Tenenbaum, Russ R. Salakhutdinov, Modelling relational data using bayesian clustered tensor factorization, in: Advances in Neural Information Processing Systems, 2009, pp. 1821–1828.
[14] Dong Yu, Li Deng, Frank Seide, Large vocabulary speech recognition using deep tensor neural networks, in: Thirteenth Annual Conference of the International Speech Communication Association, 2012.
[15] Edoardo Ragusa, Paolo Gastaldo, Rodolfo Zunino, Erik Cambria, Learning with similarity functions: a tensor-based framework, Cogn. Comput. 11 (1) (2019) 31–49.
[16] Farhad Abedini, Mohammad Bagher Menhaj, Mohammad Reza Keyvanpour, An MLP-based representation of neural tensor networks for the RDF data models, Neural Comput. Appl. 31 (2) (2019) 1135–1144.
[17] Richard Socher, Danqi Chen, Christopher D. Manning, Andrew Ng, Reasoning with neural tensor networks for knowledge base completion, in: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 26, Curran Associates, Inc., 2013, pp. 926–934.
[18] Fengyu Guo, Ruifang He, Di Jin, Jianwu Dang, Longbiao Wang, Xiangang Li, Implicit discourse relation recognition using neural tensor network with interactive attention and sparse learning, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 547–558.
[19] Andros Tjandra, Sakriani Sakti, Ruli Manurung, Mirna Adriani, Satoshi Nakamura, Gated recurrent neural tensor network, in: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 448–455.
[20] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, Danilo P. Mandic, et al., Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Found. Trends® Mach. Learn. 9 (4–5) (2016) 249–429.
[21] Gene H. Golub, Christian Reinsch, Singular value decomposition and least squares solutions, in: Linear Algebra, Springer, 1971, pp. 134–151.
[22] Hervé Bourlard, Yves Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biol. Cybernet. 59 (4–5) (1988) 291–294.
[23] Minh-Thang Luong, Hieu Pham, Christopher D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1412–1421.
[24] Dzmitry Bahdanau, Kyung Hyun Cho, Yoshua Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015, Conference date: 07-05-2015 Through 09-05-2015, 2015.
[25] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.
[26] Licheng Yu, Mohit Bansal, Tamara Berg, Hierarchically-attentive RNN for album summarization and storytelling, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 966–971.
[27] Dichao Hu, An introductory survey on attention mechanisms in NLP problems, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2019, pp. 432–448.
[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[29] Wei Li, Kun Guo, Yong Shi, Luyao Zhu, Yuanchun Zheng, DWWP: Domain-specific new words detection and word propagation system for sentiment analysis in the tourism domain, Knowl.-Based Syst. 146 (2018) 203–214.
[30] Yukun Ma, Khanh Linh Nguyen, Frank Xing, Erik Cambria, A survey on empathetic dialogue systems, Inf. Fusion 64 (2020) 50–70.
[31] W. Li, L. Zhu, Y. Shi, K. Guo, E. Cambria, User reviews: Sentiment analysis using lexicon integrated two-channel CNN-LSTM family models, Appl. Soft Comput. 94 (2020) 106435.
[32] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, Louis-Philippe Morency, Context-dependent sentiment analysis in user-generated videos, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 873–883.
[33] Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, Erik Cambria, DialogueRNN: An attentive rnn for emotion detection in conversations, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 6818–6825.
[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[35] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arxiv preprint arXiv:1412.6980.
[36] X. Zhong, E. Cambria, A. Hussain, Extracting time expressions and named entities with constituent-based tagging schemes, Cogn. Comput. 12 (4) (2020) 844–862.

[37] Qi Zhang, Jinlan Fu, Xiaoyu Liu, Xuanjing Huang, Adaptive co-attention network for named entity recognition in tweets, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018, pp. 5674–5681.
[38] Xuezhe Ma, Eduard Hovy, End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1064–1074.
[39] Erik F. Tjong Kim Sang, Fien De Meulder, Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition, in: Proceedings of the Seventh Conference on Natural Language Learning At HLT-NAACL 2003 - Volume 4, CONLL '03, Association for Computational Linguistics, USA, 2003, pp. 142–147.
[40] Jason P.C. Chiu, Eric Nichols, Named entity recognition with bidirectional LSTM-CNNs, Trans. Assoc. Comput. Linguist. 4 (2016) 357–370.
[41] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[42] John D. Lafferty, Andrew McCallum, Fernando C.N. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 282–289.
[43] George A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[44] Erik Cambria, Yang Li, Frank Z. Xing, Soujanya Poria, Kenneth Kwok, SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 105–114.
[45] Jens Graupmann, Ralf Schenkel, Gerhard Weikum, The SphereSearch engine for unified ranked retrieval of heterogeneous XML and web documents, in: Proceedings of the 31st International Conference on Very Large Data Bases, VLDB Endowment, 2005, pp. 529–540.
[46] Quan Wang, Bin Wang, Li Guo, Knowledge base completion using embeddings and rules, in: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, AAAI Press, 2015, pp. 1859–1865.
[47] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, S. Yu Philip, A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[48] Erik Cambria, Yunqing Xia, Amir Hussain, Affective common sense knowledge acquisition for sentiment analysis, in: LREC, Istanbul, 2012, pp. 3580–3585.
[49] Erik Cambria, Amir Hussain, Catherine Havasi, Chris Eckl, Common sense computing: From the society of mind to digital intuition and beyond, in: Julian Fierrez, Javier Ortega, Anna Esposito, Andrzej Drygajlo, Marcos Faundez-Zanuy (Eds.), Biometric ID Management and Multimodal Communication, in: Lecture Notes in Computer Science, vol. 5707, Springer, Berlin Heidelberg, 2009, pp. 252–259.
[50] Joseph Turian, Lev Ratinov, Yoshua Bengio, Word representations: a simple and general method for semi-supervised learning, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 384–394.
[51] José Luis Morales, A numerical study of limited memory BFGS methods, Appl. Math. Lett. 15 (4) (2002) 481–487.
[52] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee, Shrikanth S. Narayanan, IEMOCAP: Interactive emotional dyadic motion capture database, Lang. Resour. Eval. 42 (4) (2008) 335.
[53] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, Rada Mihalcea, MELD: A multimodal multi-party dataset for emotion recognition in conversations, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 527–536.
[54] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, Jamie Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, pp. 1247–1250.
[55] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer, Neural architectures for named entity recognition, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, San Diego, California, 2016, pp. 260–270.