Automatic Construction of Tensor Product Variable Binding Neural Networks for Neural-Symbolic Intelligent Systems

Alexander Demidovskij
Computer Science Department
Higher School of Economics
Nizhny Novgorod, Russia
ORCID: 0000-0003-3605-6332

***Abstract*— One of the main challenges in building hybrid systems that integrate symbolic and sub-symbolic levels of computations is elaboration of methods that would allow expressing symbolic manipulations in a form of neural networks dynamics. In particular, it is extremely important to be able to process symbolic structures in a distributed format and perform analysis over this tensor representation that would be equivalent to a sequence of symbolic operations over the original structure. In this paper, the novel method is proposed for processing and analysis of semantics tree in a form of generated neural network. This method is built upon the Tensor Product Representations (TPRs) framework, scales well for an arbitrary structure and allows expressing simple symbolic operations in a form of neural network dynamics.**

Keywords*— *sub-symbolic computations, artificial neural networks, tensor product variable binding

I. INTRODUCTION

Symbolic and sub-symbolic approaches to computations have been considered as competing paradigms for a long period of time. Symbolic level is defined by methods that manipulate symbols and explicit representations. Sub-symbolic paradigm or as it is often referred to as connectionism defines massively parallel computations and is nowadays strongly associated with Artificial Neural Networks (ANNs) [1], [2]. However, there seems to be a considerable drawback of neural computation that is low interpretability. At the same time intermediate calculation results of symbolic methods are quite transparent and easy to work with. Therefore, there is a strong intention to build integrated systems on top of both computational paradigms and produce robust and flexible solutions [3], [4].

At the same time, symbols are natural parts of any language. Language is considered to be a combinatorial system that operates with various symbols [5]: phonemes, morphemes, phrases etc. Those symbols are building bricks of arbitrary structures that appear after syntax parsing, semantic interpretation etc. Resulting symbolic structures become inputs and outputs of traditional NLP (Natural Language Processing) methods. In this paper the task of voice extraction is considered [6]. There are two sample sentences: *A homework is done by Peter* and *Peter does a homework*.

The reported study was funded by RFBR, project number 19-37-90058

After the semantic interpretation of these sentences we get the structure shown on Fig. 1 and Fig. 2, where *A* stands for *"agent"* and *P* for *"patient"*. Both *A* and *P* represent sub-trees. It is obvious that presence of *Aux* symbol as the left-child-of-left-child-of-right-child-of-root can be used as a marker of Passive voice in the sentence.

The central idea of this paper is elaboration of such a

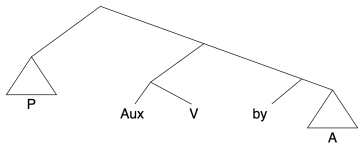

Figure 1. Parsed structure of Passive voice sentence.

neural method that is capable of performing a simple symbolic task such as retrieval of the voice marker and overall

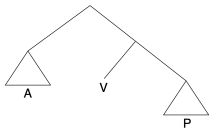

Figure 2. Parsed structure of Active voice sentence.

classification task on the neural level. At the same time, the framework of research allows generalization of the neural approach to solving other simple symbolic tasks, like linguistic assessments aggregation during decision making. In order to do that, the structure should be transferred to the distributed representation as neural networks do not work with symbols directly.

The structure of this work is as follows: Section 2 gives overview of the background study, Section 3 contains deep analysis of primitive symbolic operations that can be expressed in a neural form. In Section 4 the proposed neural network is described. Evaluation of the method is demonstrated in Section 5. Conclusions and directions of further research are formulated in the final part of the paper.

II. RELATED WORK

One of the most considerable theoretical approaches that is closely related to the discourse of this research was

978-1-7281-7116-6/20/$31.00 ©2020 IEEE

proposed in [6] and called Active-Passive Net. The overall architecture allows performing extraction of required particles of the structure and construction of the new structure that represents predicate-calculus expression. Active-Passive Net as well as the neural network proposed in current research rely on the mechanism of Tensor Product Representations (TPRs) [7]. There are several recent advances in the field that allow building symbolic and sub-symbolic integrated solutions with TPRs. The integrated symbolic and sub-symbolic flow consists of following steps (Fig. 3):

- encoding the symbolic structure as a distributed representation with a neural network. In [8] a new encoder design was proposed for the simple structure that has only one nesting level.
- flattenning the distributed representation to a vector format with a neural network [9].
- performing domain specific analysis of the structure on the neural level. For example, identification of whether the sentence is in Active or Passive voice.
- structural manipulations on the neural level, for example, joining of two trees in one [9].
- decoding the new structure from a distributed representation to symbolic level.

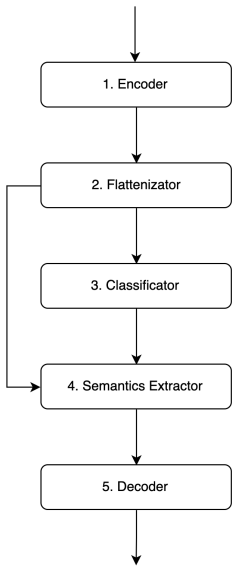

Figure 3. Desired integrated pipeline of symbolic and sub-symbolic computations.

Regarding the applied methods, various TPRs-based systems were introduced in past years. One of them is TPGN that uses TPRs in a task of image captioning [10]. Another recent approach called ROLE was proposed as an interpretation engine of distributed representations that were learned in deep neural networks [11]. This brings the question of using this apparatus for performing natural language structural analysis in the form of neural networks dynamics, for example analysis of parse trees or semantic structures of sentences. Finally, TPRs are used in applied tasks, like image captioning [12] or question answering [13].

III. Tensor Product Variable Binding

Tensor Product Variable Binding is a way to transform the symbolic structure into the vector format [14] using the tensor product operation (1).

Definition 1. Filler – a particular instance of the given structural type.

Definition 2. Role – a function that filler presents in a structure.

Definition 3. Tensor multiplication is an operation over two tensors a with rank x and b with rank y that produces a tensor z has rank $x + y$ and it consists of pair-wise multiplications of all elements from x and y.

Definition 4. Tensor product for a structure. A structure is perceived as a set of pairs of fillers $\{f_i\}$ and roles $\{r_i\}$ and its tensor product is found as (1).

$$\psi = \sum_i f_i \otimes r_i \tag{1}$$

The specific operation *ex* [7] is defined in order to extract the sub-tree by a role it plays in the original structure. The TPVB approach defines method of constructing a shift matrix W_{ex} that performs extraction of structural elements by using the distributed representation of the original structure. Shift matrix is defined with a recursive formula (5).

Definition 5. Joining operation $cons(p, q)$ is an action over two structures (trees) so that the tree p is sliding as a whole "down to the left" so that its root is moved to the left-child-of-the-root position and tree q is sliding "down to the right".

Operation *cons* can be expressed for binary trees as:

$$\begin{aligned} cons(p,q) &= p \otimes r_0 + p \otimes r_1 \\ cons_0(p) &\equiv cons(p, \emptyset) \\ cons_1(q) &\equiv cons(\emptyset, q) \end{aligned} \tag{2}$$

where r_0 and r_1 are roles, $\emptyset$ is empty tree.

It was proved [9] that this operation can be expressed in matrix form given that it operates over the tensor representation of structures (3).

$$cons(p,q) = W_{cons0}p + W_{cons1}q \tag{3}$$

Definition 6. Extracting operation $ex_X(p)$ is an action over a single structure (tree) so that the X_{th} child of the root becomes an independent tree and the remaining part of the original tree is no longer used.

Operation ex_X can be expressed for binary trees as:

$$\begin{aligned} ex_0(s) &= p \\ ex_1(s) &= q \end{aligned} \tag{4}$$

where $s = cons(p, q)$.

It was proved [9] that this operation can be expressed in matrix form given that it operates over the tensor representation of a structure (8).

$$\begin{aligned} ex_X(s) &= W_{ex_X}s \\ W_{ex_0} &= I \otimes 1_A \otimes u_0 \end{aligned} \tag{5}$$

where *I* is the identity matrix on the total role vector space, 1_A is the identity matrix. W_{ex} matrices are defined in the manner similar to the W_{cons} matrices that join two sub-trees in one structure [9]. Extraction mechanism is central to the task of defining the voice of the sentence as it was stated before (Section 1). The way the proposed neural architecture uses this extraction methodology is described in Section 4.

IV. Proposed neural design of classification TPVB network

A. Network architecture

Current research is targeted to propose a novel design for the TPVB Clasification network. It is demonstrated on Fig. 4. Each block is considered separately below.

a) Network inputs. The network accepts 1 variable input and 3 constant ones. Single variable input is a vector that represents the distributed representation of the variable structure that is flattened to a raw vector format.

b) Shift block. From the architectural point of view the massive part of the network is a set of consecutive *Shift Blocks*. The first input of the *Shift Block* is a constant input of the network that is prepared in advance during network generation on the base of vectors dual to the roles vectors (5). The second input is a structure representation at the current nesting level. This input is then used for extraction of a sub-tree by the particular role. Another considerable implementation aspect of the *Shift Block* is the chain of tensor manipulation layers. In particular, cascade of *Reshape*, *Crop3D* and *Reshape* layers. The main reason for those manipulations is technical limitations of available Keras primitives [15]. Extraction matrix is quite specific as it accepts as an input not just the flattened representation of the structure, but a reduced version of representation without components representing level 0 in the tensor representation of the whole structure. However, available *Crop3D* primitive requires tensor of a rank higher than the variable input. That is why there are additional reshaping layers to satisfy software requirements. Output of the *Shift Block* (Fig. 5) stands for the distributed representation of the sub-tree of the input structure that was binded with the specific role vector. Number of shift blocks hugely depends on the maximum depth of the input structure and on what exact part of the given structure should be retrieved. In this particular case, the maximum depth of the tree is 3 and the element to be retrieved is *Aux*.

c) Global Pooling layer. In the circumstances of the given task, the input of this layer is a plain vector. It represents extracted filler and can either contain at least one non-zero element (local representation) or no non-zero elements at all. The latter case means that there is no matching filler in the structure and the sentence is of Active voice. Due to these facts, the *GlobalMaxPooling1* primitive is used and it produces a single number as an output that represents the biggest value in the row.

d) Mask Lambda layer. Finally, the *Mask* layer accepts a single number as an input and should produce a boolean output. Keras framework exposes *Mask* functionality that allows to perform conditional operations inside the neural network. The condition is very simple: in case the input is a non-zero element, the sentence is of a Passive voice and the layer and the Network should return "one". Otherwise, the sentence is built with the use of Active voice.

B. Network generation

As it was already highlighted, the proposed design of the neural network does not imply any kind of training and is a one-shot generated architecture. In order to generate this type of networks the following inputs are required:

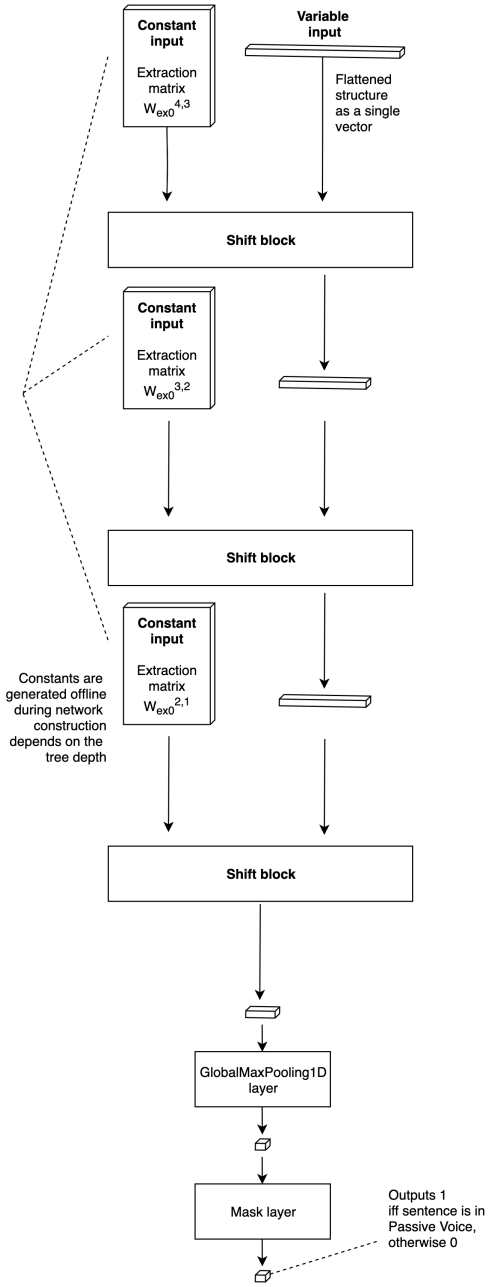

Figure 4. Proposed architecture of TPVB Classification network.

- maximum depth of the tree and dimensionality of fillers. It is required to know how deep is the tree or, more specifically, what is the maximum rank of the distributed representation of an arbitrary structure.
- set of roles vectors. Extraction operation (5) cannot be performed with dual role vectors.
- extraction rules. According to the task considered in this paper, definition of sentence voice depends on existence of leaf (*Aux*). In order to do that in a neural network several consequent shifting matrix should be applied to the input tensor, each of them corresponds

of extraction of a sub-tree that is present at a particular role.

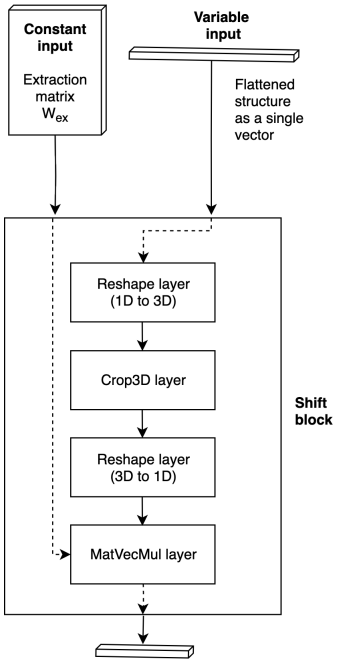

Figure 5. Proposed design of a single shifting block.

Finally, it is important to mention that there are two distinct steps in using the proposed approach. The first one is one-time generation of the network depending on the size of structures, roles and fillers vectors. Once the network is generated, it can be used multiple times for inference and only requires the distributed representation of input structure as input. The source code for further experiments can be found at the open source repository[1].

V. Evaluation of the Method Proposed

In order to explain the dynamics of the neural network, the following example is taken. Lets consider two sample sentences. The first one is *A homework is done by Peter*. The second is *Peter does a homework*. Firstly, semantic trees are created after processing of these sentences. The first sentence is split in several parts: <Patient> replaces <A homework>, <Aux> replaces <is>, <Verb> goes for <done>, <by> is left unchanged, <Agent> replaces <Peter>. The second sentence is split in three parts: <Agent> replaces <Peter>, <Verb> stands for <does>, <Patient> is for <a homework>. It is obvious that these sentences become structures from Fig. 1 and Fig. 2 correspondingly. From this moment on, symbolic structures representing two sentences become central elements of the system. The next step is to encode each structure in a tensor form. For that each filler and role are defined as linearly independent vectors.

$$
\begin{aligned}
A &= [7\ 0\ 0\ 0\ 0] \\
V &= [0\ 4\ 0\ 0\ 0] \\
P &= [0\ 0\ 2\ 0\ 0] \\
Aux &= [0\ 0\ 0\ 5\ 0] \\
by &= [0\ 0\ 0\ 0\ 3] \\
r_0 &= [10\ 0] \\
r_1 &= [0\ 5]
\end{aligned} \tag{6}
$$

According to the rules defined in [14] and using the lightweight binding network proposed in [8], [9], both sentences are translated to the distributed representation, an example of passive and active voice sentences representation are demonstrated in (7) and (8) correspondingly.

$$
\begin{aligned}
\boldsymbol{S_{Passive}} = \\
&[\mathbf{0\ 0\ 0\ 0\ 0}], \\
&\big[[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{20},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}]\big], \\
&\Big[\big[[0,0],[0,0]\big],\big[[0,0],[0,0]\big],\big[[0,0],[0,0]\big]\Big], \\
&[\Big[\big[[0,0],[0,0]\big],\big[[0,200],[0,0]\big],\big[[0,0],[0,0]\big]\Big], \\
&\Big[\big[[0,0],[0,0]\big],\big[[0,0],[0,0]\big],\big[[0,0],[0,875]\big]\Big], \\
&\Big[\big[[0,0],[0,0]\big],\big[[0,200],[0,0]\big],\big[[0,1000],[0,0]\big]\Big], \\
&\Big[\big[[0,2500],[0,0]\big],\big[[0,200],[0,0]\big],\big[[0,0],[0,0]\big]\Big], \\
&\Big[\big[[0,0],[0,750]\big],\big[[0,200],[0,0]\big],\big[[0,0],[0,0]\big]\Big]]
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
\boldsymbol{S_{Active}} = \\
&[\mathbf{0\ 0\ 0\ 0\ 0}], \\
&\big[[\mathbf{70},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}]\big], \\
&[\big[[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}]\big],\big[[\mathbf{0},\mathbf{200}],[\mathbf{0},\mathbf{0}]\big],\big[[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{50}]\big], \\
&\big[[0,0],[0,0]\big],\big[[0,0],[0,0]\big]]
\end{aligned} \tag{8}
$$

Each of these tensors is flattened and fed into the classification network as a plain vector. In order to execute the first shift block it is required to know what is the depth of the original structure. In other words the neural network has to extract the child of the root of the encoded tree. Within the Tensor Product Variable Binding framework, the depth of the maximum structure should be known in advance.

The output of the first shift block for the passive and active voices are shown in (9), (10) respectively.

$$
\begin{aligned}
\boldsymbol{S_{Passive}} = \\
&[\mathbf{0\ 0\ 0\ 0\ 0}], \\
&\big[[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}]\big], \\
&[\big[[\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0}]\big],\big[[\mathbf{0},\mathbf{200}],[\mathbf{0},\mathbf{0}]\big],\big[[\mathbf{175},\mathbf{0}],[\mathbf{0},\mathbf{200}]\big], \\
&\big[[0,0],[0,500]\big],\big[[0,0],[0,0]\big]]
\end{aligned} \tag{9}
$$

[1] https://github.com/demid5111/ldss-tensor-structures

$$\boldsymbol{S_{Active}} = \\ [\mathbf{0\ 0\ 0\ 0\ 0}], \\ \big[[\mathbf{10}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}]\big], \\ \big[\big[[\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}]\big], \big[[\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}]\big], \big[[\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}]\big], \\ \big[[0, 0], [0, 0]\big], \big[[0, 0], [0, 0]\big]\big] \tag{10}$$

The output of the second shift block for the passive and active voices are shown in (11), (12) respectively.

$$\boldsymbol{S_{Passive}} = \\ [\mathbf{0\ 0\ 0\ 0\ 0}], \\ \big[[\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{20}, \mathbf{0}]\big], \tag{11}$$

$$\boldsymbol{S_{Active}} = \\ [\mathbf{0\ 0\ 0\ 0\ 0}], \\ \big[[\mathbf{0}, \mathbf{4}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}], [\mathbf{0}, \mathbf{0}]\big], \tag{12}$$

The output of the third shift block for the passive and active voices are shown in (13), (14) respectively.

$$\boldsymbol{S_{Passive}} = [\mathbf{0\ 0\ 0\ 5\ 0}] \tag{13}$$

$$\boldsymbol{S_{Active}} = [\mathbf{0\ 0\ 0\ 0\ 0}] \tag{14}$$

It is obvious that from one shift block to another the dimensionality of data tensor reduces as well as reduces the depth of the structure that it represents. Each shift block performs extraction logic that allow retrieval of a filler on a particular position that is a key criterion for defining whether the sentence is in Active Voice or not.

VI. Conclusion

A novel method of generating the neural network for solving the symbolic task was proposed in the paper. The elaborated approach demonstrates the viability of building sub-symbolic systems capable of performing various symbolic algorithms. As for the directions of further research, neural architecture of the classification network can be used for further implementation of the Active-Passive network. It represents the integrated approach starting with the distributed tensor representation of the given structure and producing distributed representation of the structure that stands for the predicate-calculus expression. In overall the paper contributes to the development of monolithic neural-symbolic systems.

References

[1] Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. Parallel distributed processing: Explorations in the microstructure of cognition, 1(45-76), 26.

[2] Pinkas, G. (1995). Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. Artificial Intelligence, 77(2), 203-247.

[3] McMillan, C., Mozer, M. C., & Smolensky, P. (1992). Rule induction through integrated symbolic and subsymbolic processing. In Advances in neural information processing systems (pp. 969-976).

[4] Besold, T. R., Garcez, A. D. A., Bader, S., Bowman, H., Domingos, P., Hitzler, P., ... & de Penning, L. (2017). Neural-symbolic learning and reasoning: A survey and interpretation. arXiv preprint arXiv:1711.03902.

[5] Cho, P. W., Goldrick, M., & Smolensky, P. (2017). Incremental parsing in a continuous dynamical system: Sentence processing in Gradient Symbolic Computation. *Linguistics Vanguard*, *3*(1).

[6] Legendre, G., Miyata, Y., & Smolensky, P. (1991). Distributed recursive structure processing. In Advances in Neural Information Processing Systems (pp. 591-597).

[7] Smolensky, P., & Legendre, G. (2006). The harmonic mind: From neural computation to optimality-theoretic grammar (Cognitive architecture), Vol. 1. MIT press.

[8] Demidovskij, A. (2019, September). Implementation Aspects of Tensor Product Variable Binding in Connectionist Systems. In Proceedings of SAI Intelligent Systems Conference (pp. 97-110). Springer, Cham.

[9] Demidovskij, A. V. (2019, October). Towards Automatic Manipulation of Arbitrary Structures in Connectivist Paradigm with Tensor Product Variable Binding. In International Conference on Neuroinformatics (pp. 375-383). Springer, Cham.

[10] Huang, Q., Smolensky, P., He, X., Deng, L., & Wu, D. (2017). Tensor product generation networks for deep NLP modeling. arXiv preprint arXiv:1709.09118.

[11] Soulos, P., McCoy, T., Linzen, T., & Smolensky, P. (2019). Discovering the compositional structure of vector representations with role learning networks. arXiv preprint arXiv:1910.09113.

[12] Huang, Q., Smolensky, P., He, X., Deng, L., & Wu, D. (2017). Tensor product generation networks for deep NLP modeling. arXiv preprint arXiv:1709.09118.

[13] Palangi, H., Smolensky, P., He, X., & Deng, L. (2018, April). Question-answering with grammatically-interpretable representations. In Thirty-Second AAAI Conference on Artificial Intelligence.

[14] Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artificial intelligence, 46(1-2), 159-216.

[15] Chollet, F. Keras: Deep Learning framework – URL: https://keras.io.