Pat-in-the-loop: Syntax-based Neural Networks with Activation Visualization and Declarative Control

Fabio Massimo Zanzotto[1][0000−0002−7301−3596], Dario Onorati[1], Pierfrancesco Tommasino[1], Andrea Santilli[1], Leonardo Ranaldi[2][0000−0001−8488−4146], and Francesca Fallucchi[2][0000−0001−5502−4358]

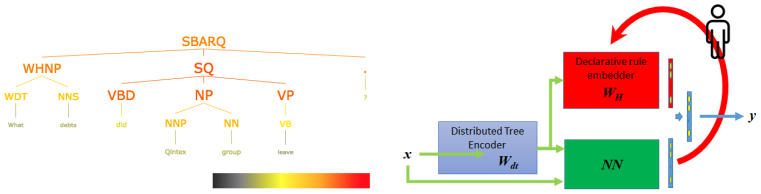
[1] ART Group, University of Rome Tor Vergata, Rome, Italy
fabio.massimo.zanzotto@uniroma2.it
[2] University Gugliemo Marconi, Rome, Italy

Abstract. The dazzling success of neural networks over natural language processing systems is imposing a urgent need to control their behavior with simpler, more direct declarative rules. In this paper, we propose Pat-in-the-loop as a model to control a specific class of syntax-oriented neural networks by adding declarative rules. In Pat-in-the-loop, *distributed tree encoders* allow to exploit parse trees in neural networks, *heat parse trees* visualize activation of parse trees, and parse subtrees are used as declarative rules in the neural network. A pilot study on question classification showed that declarative rules representing human knowledge can be effectively used in these neural networks.[3]

1 Introduction

Neural networks are obtaining dazzling successes in natural language processing (NLP). General neural networks learned on terabytes of data are replacing decades of scientific investigations by showing unprecedented performances in a variety of NLP tasks [6]. Hence, systems based on NLP and on neural networks (NLP-NN) are everywhere. As a consequence of the success, public opinion is extremely fast in spotting possibly catastrophic, unwanted behavior on deployed NLP-NN systems (see, for example, [16]). As many learned systems [4], also NLP-NN systems are exposed to biased decisions or biased production of utterances. This problem is becoming so important that extensive analyses are performed, for example, for the tricky class of systems for sentiment analysis [11]. To promptly recover from catastrophic failures, NLP-NN systems should be endowed with the possibility of modifying their behavior by using declarative languages to teach neural networks with a deductive teaching approach. Deductive teaching is an extremely difficult task even in the human learning process [1, 15]. Active learning techniques [5] can require too many examples and may focus the attention of NLP-NN systems on irrelevant peculiarities of datasets [2].

Looking into NLP-NN systems beyond the dazzling light is becoming an active area [9, 8] since traditional neural network visualization tools are obscure when applied to NLP-NN systems. Heatmaps are powerful tools for visualizing neural networks applied to image interpretation [20]. In fact, heatmaps can visualize how neural network layers treat specific subparts of images. Yet, when applied to NLP-NN systems [13] are extremely difficult to interpret.



A heat parse tree *Pat-in-the-loop*: the overall system

Fig. 2: The overall idea

In this paper, we propose *Pat-in-the-loop* as a model to include human control in specific NLP-NN systems that exploit syntactic information. The key contributions are: (1) *distributed tree encoders* that directly exploit parse trees in neural networks; (2) *heat parse trees* that visualize which parts of parse trees are responsible for the activation of specific neurons (see Figure 1a); and, (3) a declarative language for controlling the behavior of neural networks. Distributed tree encoders allow to produce heat parse trees and developers can explore activation of parse trees for specific decisions to derive rules for correcting system behavior. We performed a pilot study on question classification where *Pat-in-the-loop* showed that human knowledge can be effectively used to control the behavior of a syntactic NLP-NN system.

2 The Model

In Pat-in-the-loop (see Figure 1b), a generic developer, which we call Pat, may inspect the reasons why her/his neural network takes some decisions. In fact, Pat's neural network model is based on *distributed tree encoders* $\boldsymbol{W_{dt}}$ to directly exploit parse trees in neural networks (Sec. 2.2). Pat can visualize why some decisions are taken from the network according to parse trees of examples x by using *"heat parse trees"* (Sec. 2.1 and Sec. 2.3). Hence, Pat can control the behavior of neural networks with declarative rules represented as subtrees by encoding these rules in $\boldsymbol{W_H}$ (Sec. 2.4).

2.1 Preliminary notation

Parse trees and *heat parse trees* are core representations in our model. This section introduces the notation to describe these two representations.

Parse trees $\mathcal{T}$ and parse subtrees τ are recursively represented as trees $t = (r, [t_1, ..., t_k])$ where r is the label representing the root of the tree and $[t_1, ..., t_k]$ is the list of child trees t_i. Leaves t are represented as trees $t = (r, [])$ with an empty list of children or directly as $t = r$.

Heat parse trees, similarly to "heat trees" in biology [7], are heatmaps over parse trees (see Figure 1a). The underlying representation is an *active tree* $\bar{t}$, that is, a tree where an activation value $v_r \in \mathbb{R}$ is associated to each node: $\bar{t} = (r, v_r, [\bar{t}_1, ..., \bar{t}_k])$. Heat parse trees are graphical visualization of active trees $\bar{t}$ where colors and sizes of nodes r depend on their activation values v_r.

2.2 Distributed Tree Encoders for Exploiting Parse Trees in Neural Networks

Distributed tree encoders are the encoders used in Pat-in-the-loop to directly exploit parse trees in neural networks. These encoders, stemming from distributed tree kernels [18], give the possibility to represent parse trees in vector spaces $\mathbb{R}^d$ that embed huge spaces of subtrees $\mathbb{R}^n$. These encoders may be seen as linear transformations $\boldsymbol{W_{dt}} \in \mathbb{R}^{d \times n}$ (similarly to Johnson-Lindenstrauss Transformation [10]). These linear transformations embed vectors $\boldsymbol{x}^{\mathcal{T}} \in \mathbb{R}^n$ in the space of tree kernels in smaller vectors $\boldsymbol{y}^{\mathcal{T}} \in \mathbb{R}^d$:

$$\boldsymbol{y}^{\mathcal{T}} = \boldsymbol{W_{dt}} \boldsymbol{x}^{\mathcal{T}}$$

Columns $\boldsymbol{w_i}$ of $\boldsymbol{W_{dt}}$ encode subtree $\tau^{(i)}$ and are computed with an encoding function $\boldsymbol{w_i} = E(\tau^{(i)})$ as follows:

$$E(\tau^{(i)}) = \begin{cases} \boldsymbol{r} & \text{if } \tau^{(i)} = (r, []) \\ \boldsymbol{r} \otimes E(\tau_1^{(i)}) \otimes ... \otimes E(\tau_k^{(i)}) & \text{if } \tau^{(i)} = (r, [\tau_1^{(i)}, ..., \tau_k^{(i)}]) \end{cases}$$

where: the operation $\boldsymbol{u} \otimes \boldsymbol{v}$ is the shuffled circular convolution, that is, a circular convolution $\star$ with a permutation matrix Φ: $\boldsymbol{u} \otimes \boldsymbol{v} = \boldsymbol{u} * \Phi \boldsymbol{v}$; and, $\boldsymbol{r} \sim \mathcal{N}(0, \frac{1}{\sqrt{d}}\mathbb{I})$ is drawn from a multivariate gaussian distribution. As for tree kernels also for distributed tree encoders, linear transformations $\boldsymbol{W_{dt}}$ and vectors $\boldsymbol{x}^{\mathcal{T}} \in \mathbb{R}^n$ are never explicitly produced and encoders are implemented as recursive functions [18].

2.3 Visualizing Activation of Parse Trees

Distributed tree encoders give the possibility of using *heat parse trees* to visualize the activation of parse trees in final decisions or intermediate neuron outputs.

To compute of *active trees* $\bar{t}$ useful to produce *heat parse trees*, a neural network should be sliced at the desired layer. Let NN be the sliced neural network, $\boldsymbol{x} = \boldsymbol{x}^{\mathcal{T}}, \boldsymbol{x}^r$ and $\boldsymbol{o}$ its output: $\boldsymbol{o} = \text{NN}(\boldsymbol{W_{dt}} \boldsymbol{x}^{\mathcal{T}}, \boldsymbol{x}^r)$ where, given an example x, $\boldsymbol{x}^{\mathcal{T}}$ is the vector representing the tree $\mathcal{T}$ in the space of subtrees related to the example x, $\boldsymbol{W_{dt}}$ is the distributed tree encoder, and $\boldsymbol{x}^r$ is the rest of the features associated to x.

Our heat parse trees show the overlap of activation of subtrees in $S(\mathcal{T})$ of specific trees $\mathcal{T}$ related to a specific example x in a specific net. This shows how subtrees in $S(\mathcal{T})$ contribute to the final activation o_i, that is, a dimension of $\boldsymbol{o}$. We believe this is more convenient than representing an extremely large heatmap for the list of subtrees in $S(\mathcal{T})$ and their related value o_i.

The computation of active trees $\bar{t}$ for displaying heat parse trees is the following. The activation weight v_r of each node r represents how much the node is responsible for the activation of the overall syntactic tree for the output of the given neuron o_i. Then, the activation value v_r is computed as follows:

$$v_r = \sum_{\tau \in S(T) \text{ and } r \in \tau} \mathrm{NN}(\boldsymbol{W_{dt}} \lambda^{\frac{|\tau|}{2}} \boldsymbol{\tau}, \boldsymbol{x}_r)$$

where $\boldsymbol{\tau}$ is the one-hot vector in the subtree space that indicates the subtree τ and $r \in \tau$ detects in r is node in τ.

With the above computation of $\bar{t}$, active subtrees τ for the output o_i of a specific neuron are overlapped in single heat parse trees.

f-measure

	micro avg	*macro avg*
BoW	0.84	0.84
PureNN	0.93	0.91
HumNN	0.93	0.92

Declarative Rules

class	**rule**
ABBR	*(NP (NP (DT) (JJ full) (NN)) (PP (IN)))*
ABBR	*(SQ (VBZ) (NP) (VP (VB stand) (PP (IN for))))*
ABBR	*(NN abbrevation)*
ABBR	*(VP (VB mean))*
NUM	*(WHNP (WDT What) (NNS debts))*
NUM	*(NP (NP (NNP)(NNP)(POS))(NN))*

PureNN

	ABBR	*ENTY*	*DESC*	*HUM*	*LOC*	*NUM*
ABBR	6	0	3	0	0	0
ENTY	0	84	3	2	4	1
DESC	0	5	133	0	0	0
HUM	0	1	1	63	0	0
LOC	0	1	1	2	76	1
NUM	0	5	5	0	1	102

→

HumNN

	ABBR	*ENTY*	*DESC*	*HUM*	*LOC*	*NUM*
ABBR	7	0	2	0	0	0
ENTY	0	83	5	3	2	1
DESC	0	3	135	0	0	0
HUM	0	3	0	62	0	0
LOC	0	4	1	1	74	1
NUM	0	3	4	1	2	103

Table 1: Pat-in-the-loop's performances, discovered declarative rules and confusion matrices on QC before and after human knowledge use

2.4 Human-in-the-loop Layer

Pat has now an important possibility of understanding why decisions are taken by a specific network and, hence, s/he can define specific rules to control the

behavior of the neural network. For example, the heat parse tree in Figure 1a suggests that the subtree *(SQ,[VBD,NP,VP])* is the more active in generating the decision if this is taken for the output of a neuron that represents a final class. If Pat aims to correct the system's behavior for a given output, s/he may select the specific subtree τ and insert $E(\tau)$ as a row in matrix $\mathbf{W}_H$ that embeds *declarative rules* (see Figure 1b). This specific rule is then reused to retrain the neural network and should change decisions for examples of the same kind.

3 Pilot Experiment

We experimented with Pat-in-the-loop by using the *coarse grain* classification problem of the question classification dataset [14], which contains 5,242 training questions and 500 testing questions. The dataset is well studied. Hence, it offers a very intriguing possibility to run an experiment where a human in the loop can make the difference in calibrating the overall system.

3.1 Experimental set-up

The Pat-in-the-loop (see Figure 1b) of the experiments has the following configuration. Distributed trees $\boldsymbol{W}_{dt}\boldsymbol{x}^{\mathcal{T}}$ are encoded in a space $\mathbb{R}^d$ with $d = 4,000$. The decaying factor of tree kernels is $\lambda = 0.6$. The module $\text{NN}(\boldsymbol{W}_{dt}\boldsymbol{x}^{\mathcal{T}}, \boldsymbol{x}^r)$ is a multi-layer perceptron that combines two multi-layer perceptrons: $Synt(\boldsymbol{W}_{dt}\boldsymbol{x}^{\mathcal{T}})$ and $Sem(\boldsymbol{x}^r)$. *Synt* exploit syntactic information and its output is 1,800. *Sem* exploits a Bag-of-Word model of the input with word embedding input of 300 from fastText [3] and output of 180. *Synt* and *Sem* are concatenated and feed a multi-layer perceptron with two layers: 100 and 6. We used a ReLU activation function among layers. The last activation function is a softmax. All experiments were run for 20 epochs in Keras. Finally, we used the CoreNLP constituent-based parser [12] for parsing questions.

We compared three systems: *BoW* that contains only the word embedding used as a bag-of-word; *PureNN* that is the system without human knowledge; and *HumNN* that is the full system with Pat's declarative knowledge. We performed a 3-fold cross validation with the training set to accumulate misclassified examples for the human learning loop. Pat inspected these examples with heat parse trees and encoded some declarative rules in $\boldsymbol{W_H}$ (see Tab. 1).

3.2 Results and discussion

Results in Table 1 shows the following important facts. First, *distributed tree encoders* positively introduce syntactic information in neural networks: 0.84 to 0.93 of improvement in f-measure from *BoW* to *PureNN* (Table 1). Second, global results of the model with human knowledge (*HumNN*) are similar and even slightly higher than those of *PureNN*. Micro-average is 0.93 for both models and macro average is 0.92 for *HumNN* with respect to 0.91 of *PureNN*. Third, Pat could change the behavior of the system where he wanted. Since Pat aimed to manipulate the behavior of the system in favor of the classes *ABBR* and *NUM*,

s/he focused the attention to examples where *PlainNN* fails. Pat's rules coded in $\boldsymbol{W_H}$. After learning the new model *HumNN disturbed* by human declarative knowledge, results on the test set are encouraging. In fact, although the overall performance is unchanged, target classes have had positive improvement. Both *ABBR* and *NUM* have an additional positively classified example. This tiny improvement suggests that the model can positively use declarative human knowledge. Finally, heat parse trees are informative. In fact, Pat could understand why some specific cases were misclassified and could select declarative rules to change the behavior of the system.

Globally, results of the pilot experiment confirmed our hypothesis: human can positively manipulate the system by inducing rules from the training set.

4 Conclusions and Future Work

In the line of understanding neural networks and trying to control their behavior besides using training examples, we presented Pat-in-the-loop. Our model exploits syntactic information in neural networks by using *distributed tree encoders*, visualizes activation of syntactic information with *heat parse trees*, and encode *declarative knowledge* in a neural network. Encouraging results on a pilot study are a first *"declarative pat"* on neural networks applied to natural language processing, which may open a wide range of possible researches.

By leveraging on recent results obtained with KERMIT [19], we aim to assess results of Pat-in-the-loop and envisage novel ways to include declarative control in these specific neural networks. Endowing neural networks with declarative control may help in clarifying who is giving knowledge to these systems. In this way, we could devise machine learning models that can repay their *"teachers"* [17].

References

1. Agrusti, G., Damiani, V., Pasquazi, D., Carta, P.: Reading mathematics at school. inferential reasoning on the pythagorean theorem [leggere la matematica a scuola. percorsi inferenziali sul teorema di pitagora]. Cadmo **23**(1), 61–85 (2015). https://doi.org/10.3280/cad2015-001007
2. Allen, G.: Machine learning: The view from statistics. In: Proceedings of the AAAS Annual Meeting (2019)
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. TACL **5**, 135–146 (2017), `http://aclweb.org/anthology/Q17-1010`
4. Courtland, R.: Bias detectives: the researchers striving to make algorithms fair. Nature **558**, 357–360 (Jun 2018). https://doi.org/10.1038/d41586-018-05469-3
5. Dasgupta, S.: Analysis of a greedy active learning strategy. In: Advances in NeurIPS. MIT Press (2005), `http://papers.nips.cc/paper/2636-analysis-of-a-greedy-active-learning-strategy.pdf`
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), `http://arxiv.org/abs/1810.04805`

7. Foster, Z.S.L., Sharpton, T.J., Grünwald, N.J.: Metacoder: An R package for visualization and manipulation of community taxonomic diversity data. PLoS Computational Biology **13**(2) (2017). https://doi.org/10.1371/journal.pcbi.1005404, https://doi.org/10.1371/journal.pcbi.1005404
8. Jacovi, A., Shalom, O.S., Goldberg, Y.: Understanding Convolutional Neural Networks for Text Classification pp. 56–65 (2018). https://doi.org/doi:10.1046/j.1365-3040.2003.01027.x, http://arxiv.org/abs/1809.08037
9. Jang, K.r., Kim, S.b., Corp, N.: Interpretable Word Embedding Contextualization pp. 341–343 (2018)
10. Johnson, W., Lindenstrauss, J.: Extensions of lipschitz mappings into a hilbert space. Contemp. Math. **26**, 189–206 (1984)
11. Kiritchenko, S., Mohammad, S.: Examining gender and race bias in two hundred sentiment analysis systems. In: Proceedings of *SEM (2018), https://aclanthology.info/papers/S18-2005/s18-2005
12. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proceedings of ACL. pp. 423–430 (2003). https://doi.org/10.3115/1075096.1075150, http://dx.doi.org/10.3115/1075096.1075150
13. Li, J., Chen, X., Hovy, E., Jurafsky, D.: Visualizing and understanding neural models in nlp. In: Proceedings of NAACL (2016). https://doi.org/10.18653/v1/N16-1082, http://aclweb.org/anthology/N16-1082
14. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the COLING. ACL, Stroudsburg, PA, USA (2002). https://doi.org/10.3115/1072228.1072378, http://dx.doi.org/10.3115/1072228.1072378
15. Pasquazi, D.: Capacità sensoriali e approccio intuitivo-geometrico nella preadolescenza: Un'indagine nelle scuole. Cadmo **2020**(1), 79–96 (2020). https://doi.org/10.3280/CAD2020-001006
16. Thompson, A.: Google's sentiment analyzer thinks being gay is bad. MOTHERBOARD (Oct 2017), https://motherboard.vice.com/en_us/article/j5jmj8/google-artificial-intelligence-bias
17. Zanzotto, F.M.: Viewpoint: Human-in-the-loop artificial intelligence. J. Artif. Intell. Res. **64**, 243–252 (2019). https://doi.org/10.1613/jair.1.11345, https://doi.org/10.1613/jair.1.11345
18. Zanzotto, F.M., Dell'Arciprete, L.: Distributed tree kernels. In: Proceedings of ICML (2012)
19. Zanzotto, F.M., Santilli, A., Ranaldi, L., Onorati, D., Tommasino, P., Fallucchi, F.: Kermit: Complementing transformer architectures with encoders of explicit syntactic interpretations. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2020)
20. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: ECCV. pp. 818–833. Cham (2014)