The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Question-Answering with Grammatically-Interpretable Representations

Hamid Palangi, Paul Smolensky, Xiaodong He, Li Deng
{hpalangi,psmo,xiaohe}@microsoft.com, l.deng@ieee.org
Deep Learning Group, Microsoft Research AI
Redmond, WA *

Abstract

We introduce an architecture, the Tensor Product Recurrent Network (TPRN). In our application of TPRN, internal representations—learned by end-to-end optimization in a deep neural network performing a textual question-answering (QA) task—can be interpreted using basic concepts from linguistic theory. No performance penalty need be paid for this increased interpretability: the proposed model performs comparably to a state-of-the-art system on the SQuAD QA task. The internal representation which is interpreted is a Tensor Product Representation: for each input word, the model selects a symbol to encode the word, and a role in which to place the symbol, and binds the two together. The selection is via soft attention. The overall interpretation is built from interpretations of the symbols, as recruited by the trained model, and interpretations of the roles as used by the model. We find support for our initial hypothesis that symbols can be interpreted as lexical-semantic word meanings, while roles can be interpreted as approximations of grammatical roles (or categories) such as subject, wh-word, determiner, etc. Fine-grained analysis reveals specific correspondences between the learned roles and parts of speech as assigned by a standard tagger (Toutanova et al. 2003), and finds several discrepancies in the model's favor. In this sense, the model learns significant aspects of grammar, after having been exposed solely to linguistically unannotated text, questions, and answers: no prior linguistic knowledge is given to the model. What is given is the means to build representations using symbols and roles, with an inductive bias favoring use of these in an approximately discrete manner.

1 Introduction: Minding the gap

The difficulty of explaining the operation of deep neural networks begins with the difficulty of interpreting the internal representations learned by these networks. This problem fundamentally derives from the incommensurability between, on the one hand, the continuous, numerical representations and operations of these networks and, on the other, meaningful interpretations—which are communicable in natural language through relatively discrete, non-numerical conceptual categories structured by conceptual relations. This gap could in principle be reduced if deep neural networks were to incorporate internal representations that are directly interpretable as discrete structures; the categories and relations of these representations might then be understandable conceptually.

In the work reported here, we describe how approximately discrete, structured distributed representations can be embedded within deep networks, their categories and structuring relations being learned end-to-end through performance of a task. Applying this approach to a challenging natural-language question-answering task, we show how the learned representations can be understood as approximating syntactic and semantic categories and relations. In this sense, the model we present learns significant aspects of syntax/semantics, recognizable using the concepts of linguistic theory, after having been exposed solely to linguistically unannotated text, questions, and answers: no prior linguistic knowledge is given to the model. What *is* built into the model is a general capacity for distributed representation of structures, and an inductive bias favoring discreteness in its deployment.

Specifically, the task we address is question answering for the SQuAD dataset (Rajpurkar et al. 2016), in which a text passage and a question are presented as input, and the model's output identifies a stretch within the passage that contains the answer to the question. In our view, SQuAD provides a sufficiently demanding QA task that showing interpretability of our proposed type of distributed structural representation in a QA system that successfully addresses SQuAD provides meaningful evidence of the potential of such representations to enhance interpretability of large-scale QA systems more generally.

The proposed capacity for distributed representation of structure is provided by Tensor Product Representations, TPRs, in which a discrete symbol structure is encoded as a vector systematically built—through vector addition and the tensor product—from vectors encoding symbols and vectors encoding the roles each symbol plays in the structure as a whole (Smolensky 1990; Smolensky and Legendre 2006; Smolensky, Goldrick, and Mathis 2014). The new model proposed here is built from the BIDAF model proposed in (Seo et al. 2016) for question answering. We replace a bidirectional RNN built from LSTM units (Hochreiter and Schmidhuber 1997) with one built from TPR units; the ar-

*This work was carried out while PS was on leave from Johns Hopkins University. LD is currently at Citadel.

chitecture is called the *Tensor Product Recurrent Network*, TPRN. TPRN learns the vector embeddings of the symbols and roles, and learns which abstract symbols to deploy in which abstract roles to represent each of the words in the text-passage and query inputs.

We show how the structural roles that TPRN learns can be interpreted through linguistic concepts at multiple levels: morphosyntactic word features, parts of speech, phrase types, and grammatical roles of phrases such as subject and object. The match between standard linguistic concepts and TPRN 's internal representations is approximate.

The work reported here illustrates how learning to perform a typical natural language task can lead a deep learning system to create representations that are interpretable as encoding abstract grammatical concepts without ever being exposed to data labelled with anything like grammatical structure. It is commonly accepted among language acquisition researchers that it is in this type of setting that children typically learn their first language, so the work lends plausibility to the hypothesis that abstract notions of linguistic theory do describe representations in speakers' minds—representations that are learned in the service of performing tasks such as question-answering which (unlike, say, a parsing task) do not explicitly necessitate any such structure.

The remainder of the paper is structured as follows. Section 2 provides some background while Section 3 introduces TPR and details how it is used in the general TPRN architecture we propose here. Experimental results applying TPRN to SQuAD are presented in Section 4. The heart of the paper is Section 5 which addresses interpretation of the representations learned by TPRN. Section 6 discusses related work and Section 7 concludes.

2 The Model

The proposed TPRN architecture is built in TensorFlow (Abadi et al. 2015) on the BIDAF model proposed in (Seo et al. 2016). BIDAF is constructed from 6 layers: a character embedding layer using CNNs, a word embedding layer using GloVe vectors (Pennington, Socher, and Manning 2014), a phrase embedding layer using bidirectional LSTMs for sentence embedding (Palangi et al. 2016), an attention flow layer using a special attention mechanism, a modeling layer using LSTMs, and an output layer that generates pointers to the start and end of an answer in the paragraph. (See Fig. 1 of (Seo et al. 2016).)

TPRN replaces the LSTM cells forming the bidirectional RNN in the phrase embedding layer with recurrent TPR cells, described next: see Fig. 1.

3 TPRN: The Tensor Product Recurrent Network

This TPRN model enables the phrase-embedding layer of the model to decide, for each word, how to encode that word by selecting among $nSymbols$ symbols, each of which it can choose to deploy in any of $nRoles$ slots in an abstract structure. The symbols and slots have no meaning prior to training. We hypothesized that the symbol selected by the trained model for encoding a given input word will be interpretable in terms of the lexical-semantic content of the word (e.g., *Australia* refers to a place) while the slots will be interpretable as grammatical roles such as subject/agent, object/patient, question-restrictor phrase. In Section 5, we will test this hypothesis; we will henceforth refer to "roles" rather than "slots". In other words, our hypothesis was that the particular word tokens for which a given symbol was selected would form a lexical-semantically-related class, and the particular word tokens for which a given role was selected would form a grammatically-related class.

To function within the network, the symbols and roles must each be embedded as vectors; assume that we use vectors of dimension $dSymbols$ and $dRoles$ for symbols and roles respectively. These embedding vectors are designed by the network, i.e., they are learned during training. The network's parameters, including these embeddings, are driven by back-propagation to minimize an objective function relevant to the model's question-answering task. The objective function includes a standard cross-entropy error measure, but also *quantization*, a kind of regularization function biasing the model towards parameters which yield decisions that select, for each word, a single symbol in a single role: the selection of symbols and roles is soft-selection, and we will say that the model's encoding assigns, to the t^{th} word $w^{(t)}$, a *symbol-attention vector* $\mathbf{a}_{\mathrm{S}}{}^{(t)}$ and a *role-attention vector* $\mathbf{a}_{\mathrm{R}}{}^{(t)}$.

The quantization term in the objective function pushes towards attention vectors that are 1-hot. We do not impose this as a hard constraint because our fundamental hypothesis is that by developing *approximately* discrete representations, the model can benefit from the advantages of discrete combinatorial representations for natural language, without suffering their disadvantage of rigidity. We note that while the attention vectors are approximately 1-hot, the actual representations deployed are attention-weighted sums of fully distributed vectors arising from distributed encodings of the symbols and distributed embeddings of the roles.

In the encoding for $w^{(t)}$, the vector $\mathbf{s}^{(t)}$ encoding the symbol is the attention-weighted sum of the $nSymbols$ possible symbols: $\mathbf{s}^{(t)} = \sum_{j=1}^{nSymbols} [\mathbf{a}_{\mathrm{S}}{}^{(t)}]_j \mathbf{s}_j = \mathbf{S}\mathbf{a}_{\mathrm{S}}{}^{(t)}$ where $\mathbf{s}_j$ is the embedding of the j^{th} symbol in $\mathbb{R}^{dSymbols}$, which is the j^{th} column of the *symbol matrix* $\mathbf{S}$. Similarly, the vector encoding the role assigned to $w^{(t)}$ is $\mathbf{r}^{(t)} = \sum_{k=1}^{nRoles} [\mathbf{a}_{\mathrm{R}}{}^{(t)}]_k \mathbf{r}_k = \mathbf{R}\mathbf{a}_{\mathrm{R}}{}^{(t)}$, with $\mathbf{r}_k$ the embedding of the k^{th} symbol in $\mathbb{R}^{dRoles}$ and the k^{th} column of the *role matrix* $\mathbf{R}$. Since the symbol $\{\mathbf{s}_j\}_{j=1:nSymbols}$ and role $\{\mathbf{r}_k\}_{k=1:nRoles}$ vectors are unconstrained, they generally emerge from the learning process as highly distributed; that is even more true of the overall representations $\{\mathbf{v}^{(t)}\}$, as we now see.

The activation vector $\mathbf{v}^{(t)}$ that encodes a single word $w^{(t)}$ combines the word's symbol-embedding vector, $\mathbf{s}^{(t)}$, and its role-embedding vector, $\mathbf{r}^{(t)}$, via the outer or tensor product: $\mathbf{v}^{(t)} = \mathbf{a}_{\mathrm{S}}{}^{(t)} \mathbf{a}_{\mathrm{R}}{}^{(t)\top} = \mathbf{a}_{\mathrm{S}}{}^{(t)} \otimes \mathbf{a}_{\mathrm{R}}{}^{(t)}$. We say that $\mathbf{v}^{(t)}$ is the *tensor product representation (TPR) of the binding of sym-*

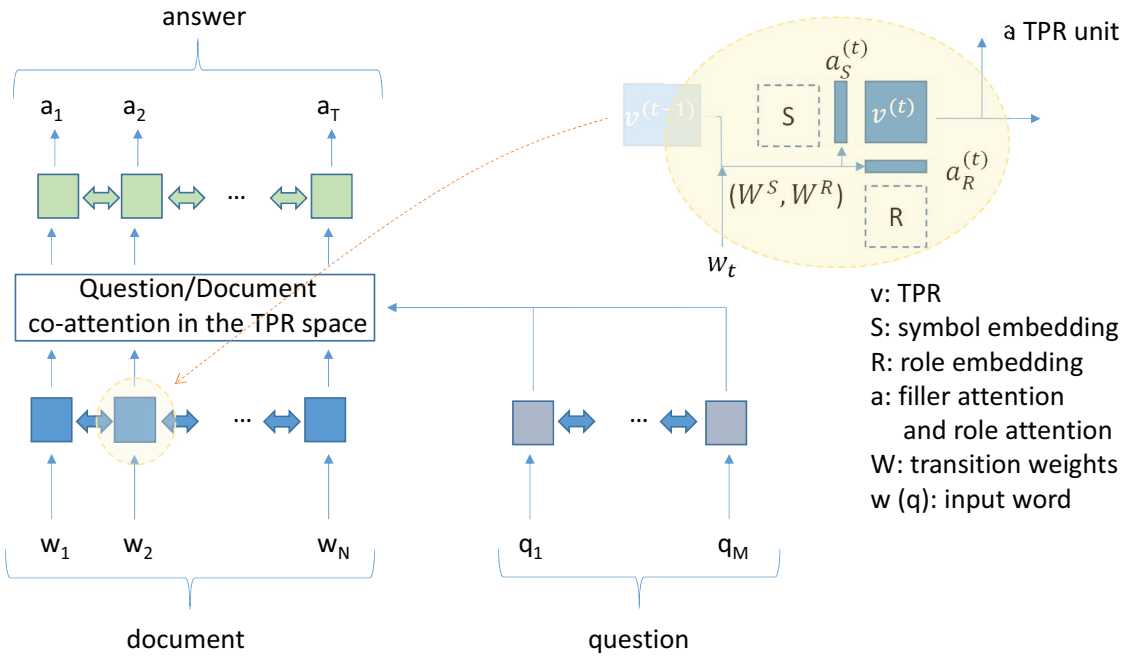

Figure 1: Block diagram of the proposed model.

bol $\mathbf{s}$ *to the role* $\mathbf{r}$. A convenient expression for $\mathbf{v}^{(t)}$ is:

$$\mathbf{v}^{(t)} \equiv \mathbf{s}^{(t)}(\mathbf{r}^{(t)})^\top = \left(\mathbf{S}\mathbf{a}_{\mathrm{S}}{}^{(t)}\right)\left(\mathbf{R}\mathbf{a}_{\mathrm{R}}{}^{(t)}\right)^\top \quad (1)$$
$$= \mathbf{S}\left(\mathbf{a}_{\mathrm{S}}{}^{(t)}\mathbf{a}_{\mathrm{R}}{}^{(t)\top}\right)\mathbf{R}^\top = \mathbf{S}\mathbf{B}^{(t)}\mathbf{R}^\top$$

The matrix $\mathbf{B}^{(t)} \equiv \mathbf{a}_{\mathrm{S}}{}^{(t)}\mathbf{a}_{\mathrm{R}}{}^{(t)\top}$ is the *binding matrix* for word $w^{(t)}$, which encodes the (soft) selection of symbol and role for $w^{(t)}$. This matrix has dimension $nSymbols \times nRoles$; the actual representation sent to deeper layers, $\mathbf{v}^{(t)}$, has $dSymbols \times dRoles$ embedding dimensions. (In the particular model discussed below, the dimensions of $\mathbf{B}$ and $\mathbf{v}^{(t)}$ are respectively 100×20 and 10×10.)

$\mathbf{a}_{\mathrm{S}}{}^{(t)}$ and $\mathbf{a}_{\mathrm{R}}{}^{(t)}$ in (1) are defined as:

$$\mathbf{a}_{\mathrm{S}}{}^{(t)} = f(\mathbf{W}^S_{in}\mathbf{w}^{(t)} + \mathbf{W}^S_{rec}vec(\mathbf{v}^{(t-1)}) + \mathbf{b}^S) \quad (2)$$
$$\mathbf{a}_{\mathrm{R}}{}^{(t)} = f(\mathbf{W}^R_{in}\mathbf{w}^{(t)} + \mathbf{W}^R_{rec}vec(\mathbf{v}^{(t-1)}) + \mathbf{b}^R) \quad (3)$$

where $vec(.)$ is the vectorization operation, $f(.)$ is the logistic sigmoid function, $\mathbf{w}^{(t)}$ is the t^{th} word and $\mathbf{b}$ is a bias vector. Equation (1) is depicted graphically in the 'TPR unit' insert in Fig. 1. During $I \rightarrow O$ computation, in the forward-directed RNN, the representation $\mathbf{v}^{(\mathbf{t-1})}$ of the previous word is used to compute the attention vectors $\mathbf{a}_{\mathrm{S}}{}^{(t)}$, $\mathbf{a}_{\mathrm{R}}{}^{(t)}$ which in turn are used to compute the representation $\mathbf{v}^{(\mathbf{t})}$ of the current word. (The same equations, with the same transition weights and biases, apply to the words in both the passage and the query.) Please note that $\mathbf{v}^{(t)}$ is initialized with zero.

Because each word is represented (approximately) as the TPR of a single symbol/role binding, we can interpret the internal representations of TPRN 's phrase-embedding layer once we can interpret the symbols and roles it has invented. Such interpretation is carried out in the Section 5.

The interest in TPR lies not only in its interpretability, but also in its power. The present TPRN model incorporates TPR to only a modest degree, but it is a proof-of-concept system that paves the way for future models that can import the power of general symbol-structure processing, proven to be within the scope of full-blown TPR architectures (Smolensky and Legendre 2006; Smolensky 2012). TPRN is designed to scale up to such architectures; design decisions such as factoring the encoding as $\mathbf{v}^{(t)} = \mathbf{a}_{\mathrm{S}}{}^{(t)}\mathbf{a}_{\mathrm{R}}{}^{(t)\top} = \mathbf{a}_{\mathrm{S}}{}^{(t)} \otimes \mathbf{a}_{\mathrm{R}}{}^{(t)}$ are far from arbitrary: they derive directly from the general TPR architecture.

As the name TPRN suggests, the novel representational capacity built into TPRN is an RNN built of TPR units: a forward- and a backward-directed RNN in each of which the word $w^{(t)}$ generates an encoding which is a TPR: $\mathbf{v}^{(t)} = \mathbf{S}\mathbf{B}^{(t)}\mathbf{R}^\top$; the binding matrix $\mathbf{B}^{(t)}$ varies across words, but a single symbol matrix $\mathbf{S}$ and single role matrix $\mathbf{R}$ apply for all words $\{w^{(t)}\}$. Both the $\mathbf{S}$ and $\mathbf{R}$ matrices are learned during training.

It remains only to specify the quantization function $\mathcal{Q}$ (4) which is added (with weight c_Q) to the cross-entropy to form the training objective for TPRN : $\mathcal{Q}$ generates a bias favoring attention vectors $\mathbf{a}_{\mathrm{S}}{}^{(t)}$ and $\mathbf{a}_{\mathrm{R}}{}^{(t)}$ that are 1-hot.

$$\mathcal{Q} = \mathcal{Q}_a(\mathbf{a}_{\mathrm{S}}{}^{(t)}) + \mathcal{Q}_a(\mathbf{a}_{\mathrm{R}}{}^{(t)}) \quad (4)$$
$$\mathcal{Q}_a(\mathbf{a}) = \Sigma_i(a_i)^2(1-a_i)^2 + \left(\Sigma_i(a_i)^2 - 1\right)^2$$

The first term of $\mathcal{Q}_a$ is minimized when each component of $\mathbf{a}$ satisfies $a_i \equiv [\mathbf{a}]_i \in \{0, 1\}$; the second term is minimized when $\|\mathbf{a}\|_2^2 = 1$. The sum of these terms is minimized when $\mathbf{a}$ is 1-hot (Cho and Smolensky 2016). $\mathcal{Q}$ drives

learning to produce weights in the final network that generate $\mathbf{a}_{\mathrm{S}}{}^{(t)}$ and $\mathbf{a}_{\mathrm{R}}{}^{(t)}$ vectors that are approximately 1-hot, but there is no mechanism within the network for enforcing (even approximately) 1-hot vectors at $I \rightarrow O$ computation (inference) time. Please note that the final loss function is the current loss function of the QA system plus the terms defined in (4).

4 Experiments

In this section, we describe details of the experiments applying the proposed TPRN model to the question-answering task of the Stanford's SQuAD dataset (Rajpurkar et al. 2016). The results of primary interest are the interpretations of the learned representations, discussed at length in the next section.

The goal of this work is not to beat the state-of-the-art system on SQuAD (at the time of writing this paper, DCN+ from Salesforce Research), but to create a high-performing question-answering system that is interpretable, by exploiting TPRs.

SQuAD is a reading comprehension dataset for question answering. It consists of more than 500 Wikipedia articles and more than 100,000 question-answer pairs about them, which is significantly larger than previous reading comprehension datasets (Rajpurkar et al. 2016). The questions and answers are human-generated. The answer to each question is determined by two pointers into the passage, one pointing to the start of the answer and the other one pointing to its end. Two metrics that are used to evaluate models on this dataset are Exact Match (EM) and F1 score.

For the experiments, we used the same settings reported in (Seo et al. 2016) for all layers of TPRN except the phrase embedding layer, which is replaced by our proposed recurrent TPRN cells. The full setting of the TPRN model for experiments is as follows:

- Questions and paragraphs were tokenized by the PTB tokenizer.
- The concatenation of word embedding using GloVe (Pennington, Socher, and Manning 2014) and character embedding using Convolutional Neural Networks (CNNs) was used to represent each word. The embedding vector for each character was of length 8 (1-D input to the CNN) and the output of the CNN's max-pooling layer over each word was a vector of length 100. The embedding size of word embedding using GloVe was also set to 100.
- For the interpretability experiments reported in Section 5, the hyperparameter values used for the TPRN cell were $nSymbols = 100$ symbols and $nRoles = 20$ roles. Embedding size was $dSymbols = 10 = dRoles$. We used $vec(\mathbf{v}^{(t)})$ as the output of our phrase embedding layer.
- The weight of the quantization regularizer in (4) was $c_Q = 0.00001$. Results were not highly sensitive to this value.
- The optimizer used was AdaDelta (Zeiler 2012) with 12 epochs.

Performance results of our model compared to the strong BIDAF model proposed in (Seo et al. 2016) are presented in Table 1. We compared the performance of single models. For the BIDAF baseline, we ran the code published in (Seo et al. 2016) with the advised hyperparameters. Similar to the LSTM used in BIDAF, we added a gating mechanism, identical to that of the LSTM cell, to the output tensor $\mathbf{v}^{(t)}$ in Equation (1). In the TPRN model tested here for performance comparison purposes, we set the number of symbols and roles to 600 and 100 respectively and the embedding size of symbols and roles to 15 and 10. Each experiment for the TPRN model took about 13 hours on a single Tesla P100 GPU. From this table we observe that our proposed TPR based model outperforms (Seo et al. 2016) by 1 point on the validation set and slightly underperforms (Seo et al. 2016) on the test set. Overall, the proposed TPRN gives results comparable to those of the state-of-the-art BIDAF model. Moreover, as we will elaborate in the following sections, our model offers considerable interpretability thanks to the structure built into TPRs.

Table 1: Performance of the proposed TPRN model compared to BIDAF proposed in (Seo et al. 2016)

Single Model	EM(dev)	F1(dev)	EM(test)	F1(test)
TPRN	63.8	74.4	66.6	76.3
BIDAF	62.8	73.5	67.1	76.8

5 Experimental interpretations of learned TPRs

We separately discuss interpretation of the symbols and the roles learned by TPRN.

5.1 Interpreting learned TPR Roles

Here we provide interpretation of the TPR roles $\mathbf{a}_{\mathrm{R}}{}^{(t)}$ assigned to the words $w^{(t)}$ of the query input in the forward-directed TPR-RNN of TPRN. (These are denoted $q^{(t)}$ in Fig. 1.) Just as good learned neural network models in vision typically acquire similar early types of representations of an input image (e.g., (Zeiler et al. 2010)), it is reasonable to hypothesize that good learned neural network models in language will typically learn low-level input representations that are generally similar to one another. Thus we can hope for some generality of the types of interpretation discussed here. Convergence on common input representations is expected because these representations capture the regularities among the inputs, useful for many tasks that process such input. The kinds of regularities to be captured in linguistic input have been studied for years by linguists, so there is reason to expect convergence between good learned neural network language-input representations and general linguistic concepts. The following interpretations provide evidence that such an expectation is merited.

We consider which word tokens $w^{(t)}$ are 'assigned to' (or 'select') a particular role k, meaning that, for an appropriate threshold θ_k, $[\hat{\mathbf{a}}_{\mathbf{R}}{}^{(t)}]_k > \theta_k$ where $\hat{\mathbf{a}}_{\mathbf{R}}{}^{(t)}$ is the L_2-normalized role-attention vector.

Grammatical role concepts learned by the model

A grammatical category—Part of Speech: Determiner $\sim$ Role #9. The network assigns to role #9 these words: a significant proportion of the tokens of: *the* (76%), *an* (52%), *a* (46%), *its* (36%) and a few tokens of *of* (8%) and *Century* (3%). The dominant words assigned to role #9 (*the, an, a, its*) are all *determiners*. Although not a determiner, *of* is also an important function word; the 3% of the tokens of *Century* that activate role #9 can be put aside. Quantitatively, $p(w \text{ is a determiner} | w \text{ activates role } \#9 \text{ to } > 0.65) = 0.96$. This interpretation does not assert that #9 is the *only* role for determiners; e.g., $p(w \text{ activates role } \#9 | w \in \{a, an, the\}) = 0.70$.

A semantic category: Predicate (verbs and adjectives) $\sim$ Role #17. The words assigned to role #17 are overwhelmingly predicates, a semantic category corresponding to the syntactic categories of verbs and adjectives [e.g., under semantic interpretation, *J runs* $\rightarrow$ *runs(J); J is tall* $\rightarrow$ *tall(J)*] . While the English word orders of these two types of predication are often opposite (*the girl runs* vs. *the tall girl*), the model represents them as both filling the same role, which can be interpreted as semantic rather than syntactic. Quantitatively, $p(w \text{ is a verb or adjective} | w \text{ selects role } \#17) = 0.82$. Unlike role #9, which concerns only a small ('closed') class of words, the class of predicates is large ('open'), and role #17 is assigned to only a rather small fraction of predicate tokens: e.g., $p(w \text{ is assigned to role } \#17 | w \text{ is a verb}) = 0.04$.

A grammatical feature: [PLURAL] $\sim$ Role #10. To observe the representational difference between the singular and plural roles we need to fix on particular words. A case study of *area* vs. *areas* revealed a total separation in their attention to role #10 (which has a midpoint level of 0.25): 100% of tokens of singular *area* have $[\hat{\mathbf{a}}_{\mathbf{R}}{}^{(t)}]_{10} < 0.25$; 100% of tokens of plural *areas* have $[\hat{\mathbf{a}}_{\mathbf{R}}{}^{(t)}]_{10} > 0.25$. This conclusion is corroborated by pronouns, where *he;him* each have mean $[\hat{\mathbf{a}}_{\mathbf{R}}{}^{(t)}]_{10} = 0.1$, while *they;them* have $[\hat{\mathbf{a}}_{\mathbf{R}}{}^{(t)}]_{10} = 0.4; 0.6$ (there are very few tokens of *she;her*).

A grammatical phrase-type: *wh*-operator restrictor 'phrase' $\sim$ Role #1. Role #1 is assigned to sequences of words including *how many teams, what kind of buildings, what honorary title*. We interpret these as approximations to a *wh-restrictor phrase*: a *wh*-word together with a property that must hold of a valid answer—crucial information for question-answering. In practice, these 'phrases' span from a *wh*-word to approximately the first following content word. Other examples are: *what was the American, which logo was, what famous event in history*.

Grammatical functions: Subject/agent vs. object/patient $\sim$ Role #6. A fundamental abstract distinction in syntax/semantics separates subjects/agents from objects/themes. In English the distinction is explicitly marked by distinct word forms only on pronouns: *he loves her* vs. *she loves him*. In the model, attention to role #6 is greater for subjects than objects, for both *he* vs. *him* and *they* vs. *them* (again, too few tokens of *she;her*). All but 13 of 124 tokens of *he* and all 77 tokens of *they* allocate high attention to #6, whereas only 1 of the 27 tokens of *him* and none of the 34 tokens of *them* do (relative to baselines appropriate for the different pairs: 0.52, 0.18).

Correcting the Stanford Tagger's POS labeling using learned roles

When *Doctor Who* is not a name: Role #7. The TV character Doctor Who (*DW*) is named many times in the SQuAD query corpus. Now in . . . *DW travels* . . . , the phrase *DW* is a proper noun ('NNP'), with unique referent, but in . . . *does the first DW see* . . ., the phrase *DW* must be a common noun ('NN'), with open reference. In such cases the Stanford tagger misclassifies *Doctor* as an NNP in 9 of 18 occurrences: see Table 2a. In . . . *the first DW serial* . . ., *first* modifies *serial* and *DW* is a proper noun. The tagger misparses this as an NN in 37 of 167 cases. Turning to the model, we can interpret it as distinguishing the NN vs. NNP parses of *DW* via role #7, which it assigns for the NN, but not the NNP, case. Of the Stanford tagger's 9 errors on NNs and 37 errors on NNPs, the model misassigns role #7 only once for each error type (shown in parentheses in Table 2a). The model makes 7 errors total (Table 2b) while the tagger makes 46. Focussing on the specific NN instances of the form *the* n^{th} *DW*, there are 19 cases: the tagger was incorrect on 11, and in every such case the model was correct; the tagger was correct in 8 cases and of these the model was also correct on 6.

When *Who* is a name: Role #1. In *Doctor Who travelled*, the word *Who* should not be parsed as a question word ('WP'), but as part of a proper noun (NNP). The Stanford tagger makes this error in every one of the 167 occurrences of *Who* within the NNP *Doctor Who*. The TPRN model, however, usually avoids this error. Recalling that role #1 marks the *wh*-restrictor 'phrase', we note that in 81% of these NNP-*Who* cases, the model does not assign role #1 to *Who* (in the remaining cases, it does assign role #1 as it includes *Who* within its *wh*-restrictor 'phrase', generated by a distinct genuine *wh*-word preceding *Who*). In all 30 instances of *Who* as a genuine question word in a sentence containing *DW*, the model correctly assigns role #1 to the question word. For example, in *Who is the producer of Doctor Who?* [query 7628], the first *Who* correctly selects role #1 while the second, correctly, does not. (The model correctly selects role #1 for non-initial *who* in many cases.)

When *to doctor* is not a verb: Role #17. The Stanford tagger parses *Doctor* as a verb in 4 of its 5 occurrences in . . . *to Doctor Who* . . .. The model does not make this mistake on any of these 5 cases: it assigns near-zero activity to role #17, identified above as the predicate role for verbs and adjectives.

Table 2: Doctor Who? Correcting the Stanford Tagger (*errors in bold*)

smallskip	**a.**		True		**b.**		True	
			NN	NNP			NN	NNP
	tagger	NN	9 (5)	**37 (1)**	**model**	NN	13 (5)	**2 (1)**
	(& model)	NNP	**9 (1)**	130 (129)	(& tagger)	NNP	**5 (1)**	165 (129)

Table 3: Symbol 27

Token	Similarity
printmaker	0.9587
composer	0.8992
who	0.8726
mathematician	0.8675
guitarist	0.8622
musician	0.8055
Whose	0.7774
engineer	0.7753
chemist	0.7485
how	0.7335
strict	0.7207

Table 4: Symbol 2

Token	Similarity
phrase	0.817
wrong	0.8146
mean	0.7972
constitutes	0.7771
call	0.7621
happens	0.752
the	0.7477
God	0.7425
nickname	0.7368
spelled	0.7162
name	0.712
happened	0.6889
as	0.6699
defines	0.647

Table 5: Symbol 6

Token	Similarity	Token	Similarity
abolished	0.8777	annexed	0.836
west	0.8734	Brisbane	0.8359
nations	0.8613	European	0.8341
Newcastle	0.8588	Scotland	0.8321
south	0.8573	Cyprus	0.8275
Melbourne	0.8558	governments	0.8266
Australia	0.8544	Commonwealth	0.8261
World	0.8526	Britain	0.8243
Belgium	0.849	flexibility	0.8227
donors	0.8476	territories	0.8219
Asian	0.8404	Switzerland	0.821
Greece	0.8402	countries	0.8206
Europe	0.8397	freedom	0.819
Thailand	0.8393	Germans	0.8178
Constituency	0.8361	north	0.8173

5.2 Interpreting learned TPR symbols

Meaning of learned symbols: Lexical-semantic coherence of symbol assignments. To interpret the lexical-semantic content of the TPR symbols $\mathbf{s}^{(t)}$ learned by the TPRN network:

1. $\mathbf{s}^{(t)} = \mathbf{S}\mathbf{a}_{\mathrm{S}^{(t)}} \in \mathbb{R}^{10}$ is calculated for all (120,950) word tokens $w^{(t)}$ in the validation set.
2. The cosine similarity is computed between $\mathbf{a}_{\mathrm{S}^{(t)}}$ and the embedding vector of each symbol.
3. The symbol with maximum (cosine) similarity is assigned to the corresponding token.
4. For each symbol, all tokens assigned to it are sorted based on their similarity to it; tokens of the same type are removed, and the top tokens from this list are examined to assess by inspection the semantic coherence of the symbol assignments (see Tables 3 – 5).

The results provide significant support for our hypothesis that each symbol corresponds to a particular meaning, assigned to a cloud of semantically-related word tokens. For example, symbol 27 and symbol 6 can be respectively interpreted as meaning 'occupation' and 'geopolitical unit'. Symbol 11 is assigned to multiple forms of the verb *to be*, e.g., *was* (85.8% of occurrences of tokens in the validation set), *is*, (93.2%) *being* (100%) and *be* (98%). Symbol 29 is selected by 10 of the 12 month names (along with other word types; more details in supplementary materials). Other symbols with semantically coherent token sets are reported in the supplementary materials. Some symbols, however, lack identifiable coherence; an example is presented in Table 4.

Polysemy. Each token of the same word, e.g., *who*, generates its own symbol/role TPR in TPRN and if our hypothesis is correct, tokens with different meaning should select different symbols. Indeed we see three general patterns of symbol selection for *who*. *Who is the producer of Dr. Who?* illustrates the main-question-word meaning and the proper-name meaning, respectively, in its two uses of *who*. Third is the relative pronoun meaning, illustrated by . . . *the actor who* . . .. The three symbol-selection patterns associated with these three meanings are shown in Table 6.

We can interpret the symbols with IDs 25, 52 and 98 as corresponding, respectively, to the meanings of a relative pronoun, a main question word, and a proper noun. The tokens with boldface counts are then correct, while the other counts are errors. Of interest are the further facts that all 18 of the non-sentence-initial main-question-word tokens are correctly identified as such (assigned symbol 52) and that, of the 27 cases of proper-noun-*who*s mislabeled with the main-question symbol 52, half are assigned role #1, placing them in the *wh*-restrictor 'phrase' (whereas only one of the 126 correctly-identified proper-noun-*who*s is). The Symbol-97-

Table 6: Symbols selected by meaning of *who*

Meaning Symbol ID:	25	52	97	98
main question word		**1062**		
relative pronoun	**14**	4	1	26
proper noun		27	16	**126**

meaning of *who* is at this point unclear.

Predicting output errors from internal misrepresentation. In processing the test query *What type/genre of TV show is Doctor Who?* [7632] the model assigns symbol 52 to *Who*, which we have interpreted as an error since symbol 52 is assigned to every one of the 1062 occurrences of *Who* as a query-initial main question-word. Although the model strongly tends to give responses of the correct category, here it replies *Time Lord*, an appropriate type of answer to a true *who* question but not to the actual question. The model makes 4 errors of this type, of the 9 errors total made when assigning symbol 25; this 44% rate contrasts with the 9% rate when it correctly assigns the 'proper-noun symbol' 98 to *Who*.

Although such error analysis with TPRN models is in its infancy, it is already beginning to reveal its potential to make it possible, we believe for the first time, to attribute overall output errors of a DNN modeling a language task to identifiable errors of internal representation. The analysis also shows how the proposed interpretations can be validated by supporting explanations for aspects of the model's behavior.

6 Related work

Architecture. In recent years, a number of DNNs have achieved notable success by reintroducing elements of symbolic computation as peripheral modules. This includes, e.g.: (i) the memory bank, a discrete set of addressed storage registers each holding a neural activation vector (Henaff et al. 2017; Sukhbaatar et al. 2015; Weston, Chopra, and Bordes 2014); and (ii) the sequential program, a discrete sequence of steps, each selected from a discrete set of simple, approximately-discrete primitive operations (Graves, Wayne, and Danihelka 2014; Neelakantan, Le, and Sutskever 2016). The discreteness in these peripheral modules is softened by continuous parameters with which they interface with the central controlling DNN; these parameters modulate (i) the writing and reading operations with which information enters and exits a memory bank ('attention' (Chorowski et al. 2015; Xu et al. 2015)); and (ii) the extent to which inputs are passed to and outputs retrieved from the set of operations constituting a program (Graves et al. 2016). The continuity of these parameters is of course crucial to enabling the overall system to be learnable by gradient-based optimization.

The present work constitutes a different approach to reintroducing approximately symbolic representations and rule-based processing into neural network computation over continuous distributed representations. In computation with TPRs, the symbols and rules are internal to the DNN; there is no separation between a central network controller and peripheral quasi-discrete modules. Items in memories are distributed representations that are combined by addition/superposition rather than by being slotted into external discrete locations. Computation over TPRs is massively parallel (Smolensky and Legendre 2006).

Interpretation. Most methods of interpreting the internal representations of DNNs do so through the input and output representations of DNNs which are by necessity interpretable: these are where the DNN must interface with our description of its problem domain. An internal neuron may be interpreted by looking at the (interpretable) input patterns that activate it, or the (interpretable) output patterns that it activates (e.g., (Zeiler and Fergus 2014)).

The method pursued in this paper, by contrast, interprets internal DNN states not via $I \rightarrow O$ behavior but via an abstract theory of the system's problem domain. In the case of a language processing problem, such theories are provided by theoretical linguistics and traditional, symbolic computational linguistics. The elements we have interpreted are TPR roles, and TPR fillers, which are distributed activation vectors incorporated into network representations via the summation of their tensor products; we have designed an architecture in which individual neurons localize the presence of such roles and fillers ($\mathbf{a}_{\mathrm{R}}{}^{(t)}$ and $\mathbf{a}_{\mathrm{S}}{}^{(t)}$). Our interpretation rests on the interrelations between activations of the roles and fillers selected to encode words-in-context with the lexical-semantic and grammatical properties attributed to those words-in-context by linguistic theories.

7 Conclusion

We introduce a modification of the BIDAF architecture for question-answering with the SQuAD dataset. This new model, TPRN , uses Tensor Product Representations in recurrent networks to encode input words. Through end-to-end learning the model learns how to deploy a set of symbols into a set of structural roles; the symbols and roles have no meaning prior to learning. We hypothesized that the symbols would acquire lexical meanings and the roles grammatical meanings. We interpret the learned symbols and roles by observing which of them the trained model selects for encoding individual words in context. We observe that the words assigned to a given symbol tend to be semantically related, and the words assigned to a given role correlate with abstract notions of grammatical roles from linguistic theory. Thus the TPRN model illustrates how learning to perform a natural language question-answering task can lead a deep learning system to create representations that are interpretable as encoding abstract grammatical concepts without ever being exposed to data labelled with anything like grammatical structure. It is widely assumed that it is in such a setting that children learn their first language, so the work lends plausibility to the hypothesis that abstract notions of linguistic theory do in fact describe representations in speakers' minds—representations that are learned in the service of performing tasks that do not explicitly necessitate any such structure.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from http://tensorflow.org/.

Cho, P. W., and Smolensky, P. 2016. Bifurcation analysis of a Gradient Symbolic Computation model of incremental processing. In Papafragou, A.; Grodner, D.; Mirman, D.; and Trueswell, J. C., eds., *Proceedings of the 38th Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Chorowski, J. K.; Bahdanau, D.; Serdyuk, D.; Cho, K.; and Bengio, Y. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, 577–585.

Graves, A.; Wayne, G.; Reynolds, M.; Harley, T.; Danihelka, I.; Grabska-Barwińska, A.; Colmenarejo, S. G.; Grefenstette, E.; Ramalho, T.; Agapiou, J.; et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471–476.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Henaff, M.; Weston, J.; Szlam, A.; Bordes, A.; and LeCun, Y. 2017. Tracking the world state with recurrent entity networks. In *6th International Conference for Learning Representations*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Neelakantan, A.; Le, Q. V.; and Sutskever, I. 2016. Neural programmer: Inducing latent programs with gradient descent. In *5th International Conference for Learning Representations*.

Palangi, H.; Deng, L.; Shen, Y.; Gao, J.; He, X.; Chen, J.; Song, X.; and Ward, R. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24(4):694–707.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*. Available at http://arxiv.org/abs/1606.05250.

Seo, M. J.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. In *5th International Conference for Learning Representations*. Available at https://arxiv.org/abs/1611.01603.

Smolensky, P., and Legendre, G. 2006. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. Cambridge, MA: The MIT Press.

Smolensky, P.; Goldrick, M.; and Mathis, D. 2014. Optimization and quantization in gradient symbol systems: A framework for integrating the continuous and the discrete in cognition. *Cognitive Science* 38:1102–1138.

Smolensky, P. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence* 46:159–216.

Smolensky, P. 2012. Symbolic functions from neural computation. *Philosophical Transactions of the Royal Society — A: Mathematical, Physical and Engineering Sciences* 370:3543–3569.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, 2440–2448.

Toutanova, K.; Klein, D.; Manning, C.; and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, 252–259. Available at https://nlp.stanford.edu/software/tagger.shtml.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, 2048–2057.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.

Zeiler, M. D.; Krishnan, D.; Taylor, G. W.; and Fergus, R. 2010. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2528–2535. IEEE.

Zeiler, M. D. 2012. ADADELTA: an adaptive learning rate method. Available at http://arxiv.org/abs/1212.5701.