Assessing the Unitary RNN as an End-to-End Compositional Model of Syntax

Jean-Philippe Bernardy Shalom Lappin
Centre for Linguistic Theory and Studies in Probability
Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg
jean-philippe.bernardy@gu.se shalom.lappin@gu.se

We show that both an LSTM and a unitary-evolution recurrent neural network (URN) can achieve encouraging accuracy on two types of syntactic patterns: context-free long distance agreement, and mildly context-sensitive cross serial dependencies. This work extends recent experiments on deeply nested context-free long distance dependencies, with similar results. URNs differ from LSTMs in that they avoid non-linear activation functions, and they apply matrix multiplication to word embeddings encoded as unitary matrices. This permits them to retain all information in the processing of an input string over arbitrary distances. It also causes them to satisfy strict compositionality. URNs constitute a significant advance in the search for explainable models in deep learning applied to NLP.

1 Introduction

[7, 10] proposed end-to-end-trainable vector space semantic models based on the syntactic types of a pregroup grammar [19]. More recently, [22] construct a vector space semantics with a modality, within a Lambek calculus-based type logical grammar.[1]

A significant inspiration for much of this work is the category theory within which quantum mechanics is formulated. The types of the Lambek calculus grammar generate a syntactic structure that is interpreted through the category of matrices. The set of such matrices constitutes a semantic representation for lexical items. They are anchored in distributional word vectors that correspond to the word embeddings of contemporary deep neural networks. The values of these matrices can be set to optimise a variety of NLP objectives, such as the evaluation of semantic distance and relatedness among sentences. Following this template, one obtains what the authors call a *compositional vector space semantics*. The compositionality of these semantic representations comes from the type system of the grammar, which is assumed as given.

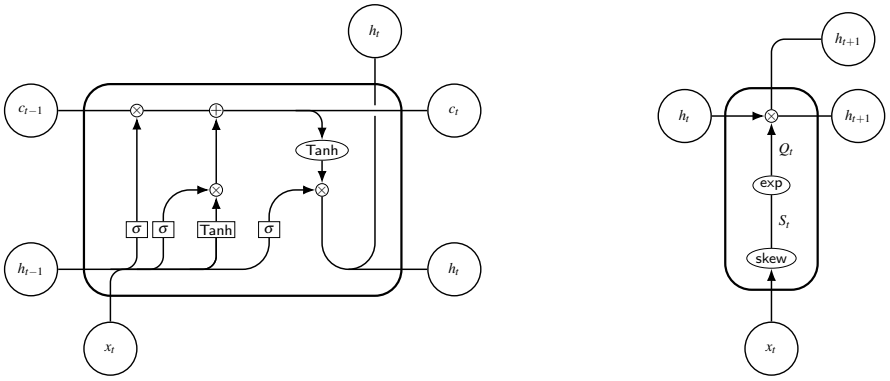
In this paper, we consider a model of compositional vector *syntax*, based on premises related to those assumed by the work just described. The main similarity with this work is that the word representations are unitary matrices, which can be trained end-to-end. The main difference is that our model does not rely on the types of an existing syntactic representation. Instead it learns syntactic structure directly from the data.

The only algebraic structure that our model invokes is that of sequences— mathematically, a free monoid over input symbols. The relevant compositional principle specifies that the representation $[\![w]\!]$ of an input sequence w is a monoid homomorphism, defined using an associative combination operator $(\cdot)$. This principle requires that the following equation holds for any two sub-sequences w_1 and w_2:

$$[\![w_1 w_2]\!] = [\![w_1]\!] \cdot [\![w_2]\!] \tag{1}$$

[1] [29] use the syntactic type representations of a Combinatory Categorial Grammar [27] to train a deep neural network to learn word representations.

M. Moortgat and G. Wijnholds: End-to-End Compositional Models of Vector-Based Semantics, 2022 (E2ECOMPVEC)
EPTCS 366, 2022, pp. 9–22, doi:10.4204/EPTCS.366.4



(a) LSTM (b) URN

Figure 1: Schematic representation of models

Hence the representation of a concatenation is the combination of representations with the operator $(\cdot)$. Additionally, the representation of the empty string must be the unit of $(\cdot)$.

In this paper, we consider the unitary-evolution recurrent neural network (URN), first suggested by [2]. This is a kind of RNN where the function applied to the hidden state vector is a unitary transformation. We analyse the capability of the URN to model two syntactic patterns: a context free structure and a mildly context sensitive one. We do this by looking at synthetic languages, which allows us to abstract from the noise that pervades natural language corpus data. We compare the syntactic capabilities of the URN with the LSTM, the dominant architecture for current RNNs.

We recap the definitions of the models and study the theoretical properties of the URN in Section 2. Our experiments are described in Sections 3 and 4. We discuss the results in Section 5, and related work in Section 6, before summarising our conclusions in Section 7.

2 Models

We consider two generative models: an LSTM [13] and an URN [2]. We portray them schematically in Fig. 1. All models are trained end-to-end as generative language models: we use cross-entropy loss for each symbol in the training set, which we sum for each position, up to the stop symbol. There is no task-specific supervised training. The training method is by stochastic gradient descent. More precisely, we use an Adam optimiser with a learning rate of 0.001, and a batch size of 512. The number of parameters for each model is listed in Tables 1 and 2.

2.1 LSTM

We provide the definition of an LSTM here to specify which version we are using, and to highlight the contrasts with URNs.

$$\begin{aligned}
v_t &= h_{t-1} \diamond x_t \\
f_t &= \sigma(W_f v_t + b_f) \\
i_t &= \sigma(W_i v_t + b_i) \\
o_t &= \sigma(W_o v_t + b_i) \\
\tilde{c}_t &= \sigma(W_c v_t + b_c) \\
c_t &= (f_t \odot c_{t-1}) + (i_t \odot \tilde{c}_t) \\
h_t &= (o_t \odot \mathsf{tanh}(c_t))
\end{aligned}$$

Here σ refers to the sigmoid function and $(\diamond)$ is vector concatenation. We apply dropout to the vectors h_t and x_t for every timestep t. Predictions are obtained by applying a projection layer to h_t, with softmax activation. The input x_t is obtained by an embedding layer.

2.2 URN

The URN, in the variant that we employ, has a much simpler definition.

$$\begin{aligned}
h_t &= Q_t h_{t-1} \\
S_t &= skew(x_t) \\
Q_t &= e^{S_t}
\end{aligned}$$

$skew(x)$ is a function that takes a vector and produces a skew-symmetric matrix by arranging the elements of x in a triangular pattern. For example, with an input vector of size 3, we have:

$$skew(x) = \begin{pmatrix} 0 & x_0 & x_1 \\ -x_0 & 0 & x_2 \\ -x_1 & -x_2 & 0 \end{pmatrix}$$

The upper triangle of S_t is provided by the previous layer (typically the word embedding layer), and its lower triangle is its negated symmetric. This setup ensures that S_t is anti-symmetric. This feature, together with the properties of matrix exponential, insures that Q_t is unitary. We call Q_t the *unitary embedding* of the input symbol at position t.

Compositional property We can now show that the URN exhibits compositionality, in the sense of Eq. (1). In general, if we let $[\![w[t]]\!] = Q_t$, then the output of the URN model for any input sequence is:

$$\begin{aligned}
h_{n+1} &= Q_n \times (Q_{n-1} \times (\cdots (Q_0 \times h_0))) \\
&= [\![w[n]]\!] \times ([\![w[n-1]]\!] \times (\cdots \times ([\![w[0]]\!] \times h_0))) \\
&= ([\![w[n]]\!] \times [\![w[n-1]]\!] \times \cdots \times [\![w[0]]\!]) \times h_0 && \text{by associativity of matrix-vector product} \\
&= [\![w[0] \ldots w[n]]\!] \times h_0
\end{aligned}$$

With:

$$[\![w[t]]\!] = e^{skew(x_t)} \qquad \text{(atomic symbol)}$$

$$[\![w_1 w_2]\!] = [\![w_2]\!] \times [\![w_1]\!] \qquad \text{(concatenation)}$$

The above definition of $[\![\_]\!]$ satisfies Eq. (1), with $P \cdot Q = Q \times P$.[2] Because unitary matrices form a monoid under product, $[\![w]\!]$ can be represented by a unitary matrix for any input string w.

Long-distance properties of URN models There are two key properties of unitary interpretations that motivate their use. First, the absence of non-linear activation functions on the recurrent path entails that they are not subject to exploding or vanishing gradients, as [2] observe. Second, because the transformations involved in their processing of input are always unitary, any difference in the start state is preserved in the output state. For every unitary matrix Q, we have

$$\langle Qh, Qs \rangle = \langle h, s \rangle$$

Another manifestation of this property is that unitary matrices can always be inverted, by transposition:[3]

$$Q^* Q = I$$

As a consequence, no information is lost through time steps. This formal analysis suggests that the URN will be good at tasks which require long-term memory. We devote the remainder of this paper to confirming experimentally that this property is observed for context-free and mildly context-sensitive inputs, when training the word embeddings by stochastic gradient descent (using the Adam optimiser).

Training regime We apply dropout to the matrix S_t for every timestep t. The input matrix S_t has dimensions $n \times n$, with n being referred as the number of units below. Predictions are obtained in the same way as for the LSTM.

3 Cross-Serial dependencies

[25] demonstrated the non-context-free nature of interleaved verb-object relations in Dutch and Swiss German. One of Shieber's examples of an embedded verb-object crossing dependency in Swiss German is given in example (A) below.

(A) Jan sät das mer **d'chind** em ***Hans*** es huus **lönd** ***hälfe*** aastriiche
Jan said that we the children-ACC Hans-DAT the house-ACC let help paint
Jan said that we let the children help Hans paint the house.

Similar patterns have been observed in other languages. They can be expressed by indexed grammars [1, 23], as well as a variety of other Mildly Context-Sensitive grammar formalisms [16, 26].

[25] observed that cross-serial dependency patterns of case marked nouns and their corresponding verbs can be iterated in this construction. The above pattern can be abstracted as a set of $a^m b^n c^m d^n$ structures, which together form a Mildly Context-Sensitive language.

Formally, we consider the family of languages $\mathscr{C}_k = \{a^m b^n c^m d^n \mid m+n < k\}$. Note that if $k < l$, then $\mathscr{C}_k \subset \mathscr{C}_l$. The training set consists of 51,200 strings picked uniformly from $\mathscr{C}_8$. The test set contains 5,120 strings picked uniformly from $\mathscr{C}_{10}$.

We recall that the RNNs are trained as generative language models. That is, assuming a sample string $w \in \mathscr{C}_{10}$, RNNs are trained to predict the symbol w_{i+1} given w_0 to w_i. Special start and a stop symbols are added to the input strings, as is standard.

[2] We leave it to the reader to check that this definition of the $(\cdot)$ operator is associative.

[3] Additionally, one should take the complex conjugate when dealing with complex-valued matrices.

Number of units	LSTM	URN
32	7314	5290
16	2738	1370
8	1218	370

Table 1: Number of parameters for cross-dependency pattern models

This is to be contrasted with the testing procedure. At test time, given the prefix $w_0 \dots w_i$, a prediction of symbol w_{i+1} is deemed correct if it is a possible continuation for $w_0 \dots w_i$; that is, if $w_0 \dots w_{i+1}$ is a prefix of some string in $\mathscr{C}_{10}$. A set of predictions for a full string $x_0 \dots x_k$ is classified as correct, if all predictions are correct up to and including the stop symbol. We report error rates for full strings only. This is because when models make a mistake, it is typically for a single symbol near the end of a string.

3.1 Results

Both RNNs struggle to generalise these patterns. They can model the training data well, but produce incorrect patterns in some cases on strings of any greater length than the samples in the training corpus.

We report four sets of results. The first set (Fig. 2a) is the cross-entropy loss for the *test set* obtained by each model across training epochs. Low losses indicate that, on a per-character basis, the models reproduce the *exact* strings in the test set, rather than making correct predictions, as defined above. We see that LSTM models generally make better guesses than URNs, across the board.

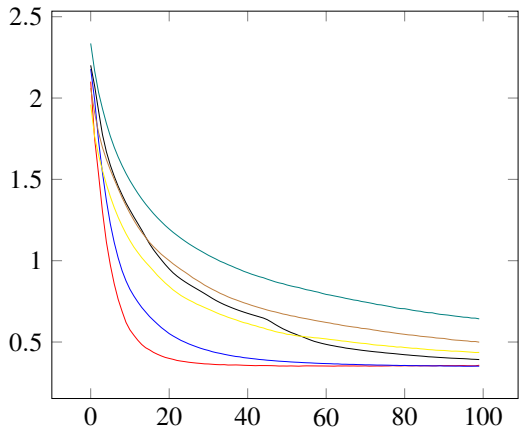
The second set (Fig. 2b) is the error rate over number of epochs, for each tested model. The best models can achieve less then ten percent error rate on average on the test set. However, the LSTM models with a larger number of units exhibit overfitting.

In the third set we show the error rate for a given training loss in Fig. 2c. The corresponding ratio is a measure of a model's capacity for correct predictions of a given quality of approximation of the training set. It can be taken as an indication of the model's bias for this task, relative to generative language modelling. The URN models tend to achieve lower error rates for this task, even though they do less well from a generative language modelling perspective. For instance, the 32-unit URN is able to obtain an error rate below 0.4 with a training loss as high as 1. In general, the URNs provide a smoother decrease in error rate as they learn the language. In contrast, the LSTM models exhibit a sharp drop in error rate around 0.7 training loss.

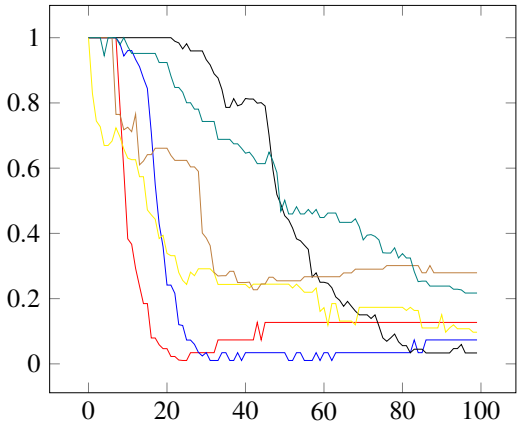
Finally, we show the error rate broken down by length of pattern (reported as $n+m$). We see here that the LSTMs tend to do better overall than the URNs for lengths unseen in the training data. However, the LSTMs do worse when the number of units increases, while the URNs do better as that number increases, thanks to a lack of overfitting.

4 Generalised Dyck Languages

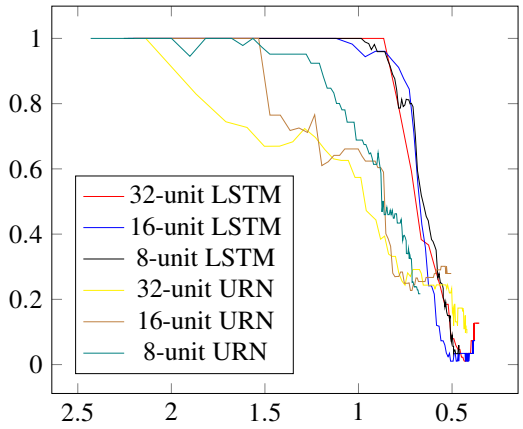
In the next experiment, we evaluate the long-distance modelling capabilities of an RNN for a context-free language. As before, we do it in a way that abstracts away from the noise of natural language, by constructing synthetic data. Following [3], we use a (generalised) Dyck language. This language is composed solely of matching parenthesis pairs. So the strings "`{([])}<>`" and "`{()[<>]}`" are part of the language, while "`<)`" is not.



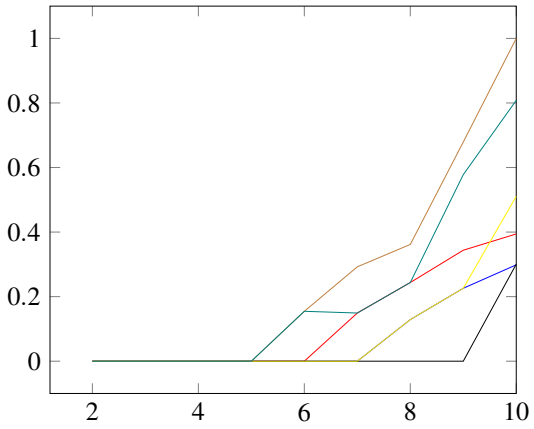
(a) Test loss across epochs



(b) Error rate across epochs



(c) Error rate against training loss.



(d) Error rate against $n+m$, at epoch 100

Figure 2: Cross-Serial dependencies results for various models

Formally, we use the language $\mathscr{D}$ defined as the set of strings generated by the following context-free rules: $E ::= \varepsilon; E ::= EE; E ::= oEc$, where (o,c) stands for a pair of matching parenthesis pairs. In all of our tests, we use 5 types of pairs (corresponding, for example, to the pairs `()` , `[]`, `{}`, `<>` and `‘’`.)

The aim of the task is to predict the correct type of *closing* parenthesis at every point in a string. It should be noted that this experiment is an idealised version of the agreement task proposed by [21]. The opening parenthesis plays the role of a word (say a noun) which governs a feature of a subsequent word (say the number of a verb), represented by the closing parenthesis. Matching of parentheses corresponds to agreement. [21] point out that sustaining accuracy over long distances requires that the model have knowledge of hierarchical syntactic structure. If an RNN captures the long-distance dependencies involved in agreement relations, it cannot rely solely on the nearby governing symbols. In particular, the accuracy must be sustained as the number *attractors* increases. For our experiment, an attractor is defined as an opening parenthesis occurring within a matching pair, but of the wrong kind. For instance, in "`{()}`", the parenthesis "`(`" is an attractor.

To generate a string with N matching pairs, we perform a random walk between opposite corners of a square grid of width and height N, such that one is not allowed to cross the diagonal. When not restricted by the boundary, a step can be taken either along the x or y axis with equal probability. A step along the x axis corresponds to opening a parenthesis, and one along the y axis involves closing one. The type of parenthesis pair is chosen randomly and uniformly.

In this task, we use strings with a length of exactly 20 characters. We train on 102400 and test on 5120 random strings. In the previous experiment we varied the string length between training and testing. In this experiment, we vary the *nesting depth*, from 3 to 9. For this purpose, we define the depth of the string is the maximum nesting level reached within it. For instance "`[{}]`" has depth 2, while "`{([()]<>)}`" has depth 4.

As in the first experiment, for the training phase the RNNs are treated as generative language models, applying a cross-entropy loss function at each position in the string. At test time, we evaluate the model's ability to predict the right kind of closing parenthesis at each point. We ignore predictions regarding opening parentheses, because they are always acceptable for the language.

Training is performed with a learning rate of 0.01, and a dropout rate of $\rho = 0.05$, for 100 epochs.

4.1 Results

We report four sets of results. The first set (Fig. 3a) is the cross-entropy loss for the *test set* achieved by each model across training epochs. Low losses indicate that, on a per-character basis, the models reproduce the *exact* strings in the test set. These losses cannot drop to zero because it is always valid to predict an opening parenthesis. As in the cross-serial task, we observe that LSTM models make better guesses than URN, at least for a similar number of units. However, we see that the training of the URN is uniformly monotonous, while the LSTM can sometimes become worse for a few epochs before converging. In fact, for 8 units, the LSTM exhibits overfitting. The test loss increases slowly after epoch 30.

To analyse the performance of each model on the task, we break down the error rate by number of attractors (Fig. 3b). The URN models are weakest for a low number of attractors, and they achieve near excellent accuracy for a large number of attractors. According to [21], this suggests that the models are highly successful in learning hierarchical structure. A high numbers of attractors corresponds to outer pairs, while a low number of attractors corresponds to inner pairs. In sum, in inner positions the URN suffers from some confusion. This confusion decreases as the number of units increases.

The LSTM is able to predict outer pairs rather well (but nowhere near as successfully as the URN

Number of units	LSTM	URN
32	6300	6348
16	2204	1644
8	924	444

Table 2: Number of parameters Dyck language models

models). It reaches almost perfect accuracy for adjacent pairs, with zero attractors, such as `[]`. It does worst on pairs with 3 to 4 attractors. The LSTM is good at making a prediction which depends only on the previous symbol. LSTM models with a larger number of units are also good at making a prediction for a pair which encloses the whole string. This indicates that it is fairly limited in its ability to capture hierarchical structures over long distances, even though it does much better than the majority class baseline, which stands at an 80% error rate.

In what follows, we will consider only the the *maximum* error rate for any given number of attractors, for varying epochs. That is, we report the peak value from the previous graph as training progresses. Using this metric, a URN performs consistently better than an LSTM with the same number of units (and a similar number of parameters). It does so consistently across the training period (Fig. 3c). However, we note that every model is capable of generalising to deeper nesting levels to some extent, with an accuracy well above a majority class baseline (with an error rate of 80%). The URN models beat the majority class baseline within the first epoch, while the LSTM needs a couple of epochs to do so.

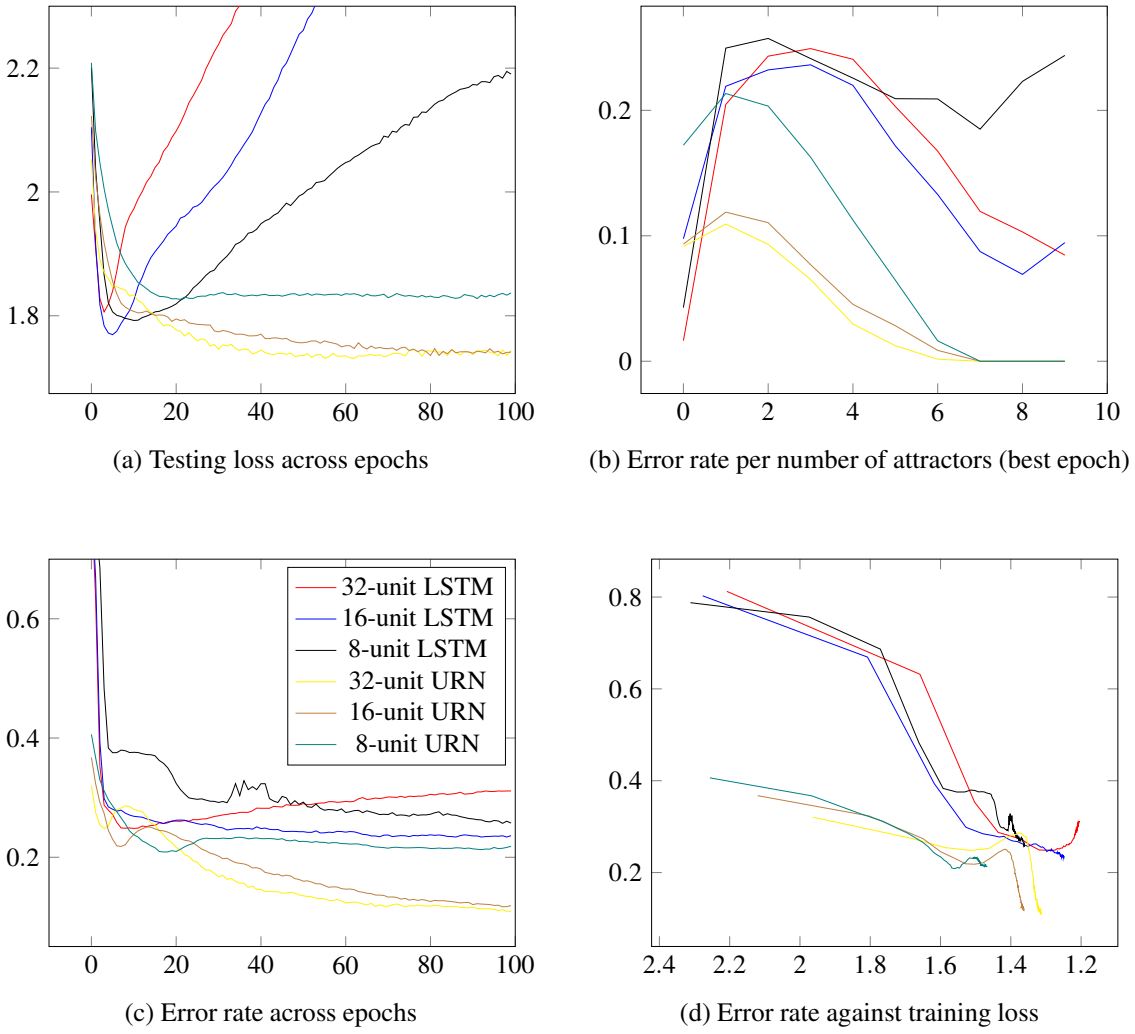
Finally we report the error rate against training loss (Fig. 3d), as we did for the cross-serial dependency task. Again, we do not report the average error rate, but rather the *maximum* error rate for any given number of attractors. Here too, the relationship between error-rate and training loss corresponds to the bias of the model for the task at hand, compared to a generative language model task. We observe that the URN models outperform the LSTM models across the board.

5 Discussion

In summary, both the URN and the LSTM perform reasonably well on our experiments. On the cross-serial dependency task, both architectures can model the training data, but they do not generalise perfectly to extended sequences. Still on both tasks, we observe that the URN exhibits a bias for the task, rather than for pure generative language model predictions. Additionally, the URN appears to be less prone to overfitting, on both tasks.

These experiments are significant for at least two reasons. We believe that the URN is the first RNN capable of recognising hierarchical syntactic structures of the sort that characterise natural language syntax. While the LSTM can capture the patterns appearing in the training set, the URN displays better generalisation capabilities than the LSTM, in addition to being mathematically tractable. A particularly attractive result, is that URN models achieve high accuracy with less than a couple of thousand parameters.

Second, our experiments illustrate the effectiveness of URNs as devices for tracking and predicting complex dependency relations, over long strings, in a fully compositional way. Unlike LSTMs, they do not suffer from opaqueness due to non-linear activation functions. They do not make use of such functions, and so their behaviour is, in principle, amenable to analysis using standard tools from linear algebra. They are not blackbox processing devices that require indirect methods of analysis and assessment, as is the case with most other deep neural networks.



(a) Testing loss across epochs

(b) Error rate per number of attractors (best epoch)

(c) Error rate across epochs

(d) Error rate against training loss

Figure 3: Dyck language experiment results. When reporting error rates as training progresses, we use the maximum error rate across number of attractors.

6 Related Work

6.1 Long-distance agreement

The capacity of Recurrent Neural Networks (RNNs), particularly LSTMs, to identify context-free long distance dependencies has been widely discussed in the NLP and cognitive science literature [8, 21, 5, 3, 24, 11, 20]. These discussions have considered dependency patterns in both artificial systems, particularly Dyck languages, and in natural languages, with subject-verb agreement providing a paradigm case.

[9] already observed that it is useful to experiment with artificial systems to filter out the noise of real world natural language data.

[3] tested the ability of the LSTM to predict closing parenthesis types in a Dyck language. The results are qualitatively similar. In both cases the LSTM makes the worse predictions for a moderate number of attractors. However he reports worse results than us, despite using an LSTM with more units. We attribute this difference to a better implementation of the LSTM. A less perspicuous application of dropouts is a likely factor in the poorer performance of [3]'s LSTM ([3]). [24] performed a set of experiments with the same goal, and reported results compatible with those of [3].

While LSTMs (and GRUs) exhibit a certain capacity to generalise to deeper nesting, their accuracy declines in relation to nesting depth. This is also the case with their handling of natural language agreement. Other experimental work has illustrated this effect [12, 24]. Similar conclusions are observed for generative self-attention architectures [31]. Significantly, recent work has indicated that non-generative self-attention architectures, in the style of BERT, simply fail at this task [4]. This suggests that sequential processing is required to solve it.

By contrast URNs achieve excellent performance on this task, without any decline in relation to either nesting depth, or number of attractors. In recent work [6] provide an explanation as to why this is the case, for a version of the URN restricted to unitary matrices acting on 3 hyperplanes. They show that the learned unitary embeddings for matching parentheses are nearly the inverses of each other: $[\![o\,c]\!] = [\![c]\!] \times [\![o]\!] \approx I$ for every pair of matching parentheses (o,c). Such an analysis illustrates the role of compositionality in the performance of URNs on an NLP task.

6.2 Cross-serial patterns

[18] study both nested and cross serial dependencies with a Simple Recurrent Network (SRN). As far as we are aware, our experiment is the first application of both LSTMs and URNs to cross serial dependency relations. While both achieve good accuracy on the cross serial patterns, the URN offers significant advantages in simplicity and transparency of architecture. It also displays enhanced stability in learning, and power of structural generalisation, relative to loss in training data.

6.3 Unitary-Evolution Recurrent Networks

[2] propose Unitary-Evolution recurrent networks to solve the problem of exploding and vanishing gradients, caused by the presence of non-linear activation functions. Despite this, [2] suggest adding ReLU activation between time-steps, unlike URNs. We are primarily concerned with the structure of the underlying unitary embeddings. The connection between the two lines of work is that if an RNN suffers exploding/vanishing gradients, it cannot track long distance dependencies.

Additionally, [2] use a complicated function to transform word representations into unitary matrices. We use a simpler method (exponential of anti-symmetric matrix), previously applied by [14] for deep

learning models. This method has performed well for the tasks discussed here. Because we use a fully general matrix exponential implementation, our model is computationally more expensive than others [2, 15]. When testing the unitary matrix encodings of [15] and [2], we obtained much worse results for our experiments. This may be because we do not include ReLU activation, while they do. Another option for enforcing the unitary character of matrices is to let back-propagation update the unitary matrices arbitrarily $n \times n$, and project them onto the unitary space periodically [30, 17].

Tensor Recurrent Neural Networks [28] describe what they call a "tensor recurrent neural network" in which the transition matrix is determined by each input symbol. This design appears to be similar to URNs. However, unlike URNs, they use non-linear activation functions, and so they inherit the complications that these functions produce.

7 Conclusions

The fact that URNs achieve good precision in predicting cross serial dependencies and generalise appropriately for nested patterns suggest that they are suitable for recognition of complex syntactic structures of the sort that are challenging for other neural networks.

Our experiments show that URNs are biased towards predicting the patterns found in context-free and mildly context-sensitive languages, even when trained as generative language models. The fact that they satisfy strict compositionality offers an important advance in the search for explainable AI systems in deep learning models.

The move to powerful bidirectional transformers, like BERT, has produced enhanced performance in a variety of NLP and other AI tasks. This has been achieved at the expense of formal grounding and computational transparency. It is even less obvious why such models perform as well as they do on some tasks, and poorly on others, than is the case for LSTMs. By contrast, URNs offer simple, light weight deep neural networks whose operation is fully open to inspection and understanding at each point in the processing regime. They can model complex syntactic structures, in the sense that they can reproduce an extensive training set of (artificial) data containing such patterns. For cross-serial dependency patterns, they do not generalise very well, but for hierarchical patterns they display impressive generalisation capabilities.

URNs provide a principled solution to the problem of syntactic compositionality. They resolve the question of how to generate the composite values of input arguments in a principled and straightforward way. This is not the case for LSTMs, because the combination of two cells cannot be expressed as a single cell. By contrast, every URN cell applies matrix multiplication to its constituents, and so the composition of the effects of two cells is simply a matrix product.

URNs are worth exploring further as models of learning and representation that bear some correspondence to human processing. In future work we will be studying the application of URNs to other cognitively interesting NLP tasks. We are particularly interested in examining possible parallels between the ways in which URNs handle linguistic information, and human understanding of natural language meaning and structure.

Finally, we observe that the unitary matrices through which URNs compute output values from input arguments are identical to the gates of quantum logic. This suggests the intriguing possibility of implementing these models as quantum circuits. At some point in the future, this may facilitate training these models on large amounts of data, and efficiently generating results for tasks that are currently beyond the resources of conventional computational systems.

In sum, we see the research that we report here as extending and modifying some of the leading ideas in the foundational work of [7, 10]. They provided a system for handling computational semantics compositionally with structures that map onto the matrices of quantum circuits. We offer a model for learning syntactic structure, and for processing in general, that is strictly compositional. It uses some of the same core methods as the earlier work. In future research we will attempt to apply our model to NLP tasks involving semantic interpretation.

Acknowledgements

The research reported in this paper was supported by grant 2014-39 from the Swedish Research Council, which funds the Centre for Linguistic Theory and Studies in Probability (CLASP) in the Department of Philosophy, Linguistics, and Theory of Science at the University of Gothenburg. We presented some of the main ideas of this paper to the CLASP Seminar, in December 2021, and to the Cognitive Science Seminar of the School of Electronic Engineering and Computer Science, Queen Mary University of London, in February 2022. We are grateful to the audiences of these two events for useful discussion and feedback.

References

[1] Alfred V. Aho (1968): *Indexed Grammars–An Extension of Context-Free Grammars*. Journal of the ACM 15(4), p. 647–671, doi:10.1145/321479.321488.

[2] Martin Arjovsky, Amar Shah & Yoshua Bengio (2016): *Unitary Evolution Recurrent Neural Networks*. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, pp. 1120–1128, doi:10.48550/arXiv.1511.06464.

[3] Jean-Philippe Bernardy (2018): *Can RNNs Learn Nested Recursion?* Linguistic Issues in Language Technology 16, doi:10.33011/lilt.v16i.1417.

[4] Jean-Philippe Bernardy, Adam Ek & Vladislav Maraev (2021): *Can the Transformer Learn Nested Recursion with Symbol Masking?* In: Findings of the ACL 2021, doi:10.18653/v1/2021.findings-acl.67.

[5] Jean-Philippe Bernardy & Shalom Lappin (2017): *Using Deep Neural Networks to Learn Syntactic Agreement*. Linguistic Issues In Language Technology 15(2), p. 15, doi:10.33011/lilt.v15i.141.

[6] Jean-Philippe Bernardy & Shalom Lappin (2022): *A Neural Model for Compositional Word Embeddings and Sentence Processing*. In: Proceedings of The Workshop on Cognitive Modeling and Computational Linguistics, Association for Computational Linguistics, doi:10.18653/v1/2022.cmcl-1.2. Available at `https://aclanthology.org/2022.cmcl-1.2/`.

[7] Bob Coecke, Mehrnoosh Sadrzadeh & Stephen Clark (2010): *Mathematical Foundations for a Compositional Distributional Model of Meaning*. Lambek Festschrift, Linguistic Analysis 36, doi:10.48550/arXiv.1003.4394.

[8] Jeffrey L. Elman (1990): *Finding structure in time*. Cognitive Science 14(2), pp. 179–211, doi:10.1016/0364-0213(90)90002-E.

[9] Jeffrey L. Elman (1991): *Distributed representations, simple recurrent networks, and grammatical structure*. Machine learning 7(2-3), pp. 195–225, doi:10.1007/BF00114844.

[10] Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke & Stephen Pulman (2011): *Concrete Sentence Spaces for Compositional Distributional Models of Meaning*. In: Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011). Available at `https://aclanthology.org/W11-0114`.

[11] Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen & Marco Baroni (2018): *Colorless Green Recurrent Networks Dream Hierarchically*. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, New Orleans, Louisiana, pp. 1195–1205, doi:10.18653/v1/N18-1108.

[12] John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang & Christopher D Manning (2020): *RNNs can generate bounded hierarchical languages with optimal memory*. *arXiv preprint arXiv:2010.07515*, doi:10.18653/v1/2020.emnlp-main.156.

[13] Sepp Hochreiter & Jürgen Schmidhuber (1997): *Long short-term memory*. *Neural Computation* 9(8), pp. 1735–1780, doi:10.1162/neco.1997.9.8.1735.

[14] Stephanie L Hyland & Gunnar Rätsch (2017): *Learning unitary operators with help from u(n)*. In: *Thirty-First AAAI Conference on Artificial Intelligence*, doi:10.1609/aaai.v31i1.10928.

[15] Li Jing, Yichen Shen, Tena Dubček, John Peurifoi, Scott Skirlo, Yann LeCun, Max Tegmark & Marin Soljačić (2017): *Tunable Efficient Unitary Neural Networks (EUNN) and their application to RNN*. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, JMLR.org, p. 1733–1741.

[16] Aravind K. Joshi, K. Vijay Shanker & David Weir (1990): *The Convergence of Mildly Context-Sensitive Grammar Formalisms*. Technical Report, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

[17] Bobak Kiani, Randall Balestriero, Yann Lecun & Seth Lloyd (2022): *projUNN: efficient method for training deep networks with unitary matrices*, doi:10.48550/arXiv.2203.05483. Available at `https://arxiv.org/pdf/2203.05483.pdf`.

[18] Christo Kirov & Robert Frank (2012): *Processing of nested and cross-serial dependencies: an automaton perspective on SRN behaviour*. *Connection Science* 24(1), pp. 1–24, doi:10.1080/09540091.2011.641939.

[19] Joachim Lambek (2008): *Pregroup Grammars and Chomsky's Earliest Examples*. *Journal of Logic, Language and Information* 17, pp. 141–160, doi:10.1007/s10849-007-9053-2.

[20] Shalom Lappin (2021): *Deep Learning and Linguistic Representation*. CRC Press, Taylor & Francis, Boca Raton, London, New York, doi:10.1201/9781003127086.

[21] Tal Linzen, Emmanuel Dupoux & Yoav Golberg (2016): *Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies*. *Transactions of the Association of Computational Linguistics* 4, pp. 521–535, doi:10.1162/tacl_a_00115.

[22] Lachlan McPheat, Mehrnoosh Sadrzadeh, Hadi Wazni & Gijs Wijnholds (2021): *Categorical Vector Space Semantics for Lambek Calculus with a Relevant Modality (Extended Abstract)*. *Electronic Proceedings in Theoretical Computer Science* 333, pp. 168–182, doi:10.4204/EPTCS.333.12.

[23] Stephen Pulman & G. D. Ritchie (1985): *Indexed Grammars and Intersecting Dependencies*. Technical Report 23, University of East Anglia.

[24] Luzi Sennhauser & Robert Berwick (2018): *Evaluating the Ability of LSTMs to Learn Context-Free Grammars*. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Association for Computational Linguistics, Brussels, Belgium, pp. 115–124, doi:10.18653/v1/W18-5414.

[25] Stuart M. Shieber (1985): *Evidence against the context-freeness of natural language*. *Linguistics and Philosophy* 8(3), pp. 333–343, doi:10.1007/BF00630917.

[26] Edward P. Stabler (2004): *Varieties of crossing dependencies: Structure dependence and mild context sensitivity*. *Cognitive Science* 93(5), pp. 699–720, doi:10.1207/s15516709cog2805_4.

[27] Mark Steedman (2000): *The Syntactic Process*. MIT Press, Cambridge, MA.

[28] Ilya Sutskever, James Martens & Geoffrey E. Hinton (2011): *Generating Text with Recurrent Neural Networks*. In Lise Getoor & Tobias Scheffer, editors: *Proceedings of the 28th International Conference on*

Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011, Omnipress, pp. 1017–1024. Available at https://icml.cc/2011/papers/524_icmlpaper.pdf.

[29] Gijs Wijnholds, Mehrnoosh Sadrzadeh & Stephen Clark (2020): *Representation Learning for Type-Driven Composition*. In: *Proceedings of the 24th Conference on Computational Natural Language Learning*, Association for Computational Linguistics, pp. 313–324, doi:10.18653/v1/2020.conll-1.24.

[30] Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux & Les Atlas (2016): *Full-capacity unitary recurrent neural networks*. *Advances in neural information processing systems* 29, pp. 4880–4888, doi:10.48550/arXiv.1611.00035.

[31] Xiang Yu, Ngoc Thang Vu & Jonas Kuhn (2019): *Learning the Dyck language with attention-based Seq2Seq models*. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 138–146, doi:10.18653/v1/W19-4815.