University of Amsterdam

MSc Computational Science
Master of Science Thesis

Pondering in Artificial Neural Networks
Inducing systematic compositionality in RNNs

Anand Kumar Singh
(11392045)

Thursday 16th August, 2018

Informatics Institute · University of Amsterdam

Pondering in Artificial Neural Networks

Inducing systematic compositionality in RNNs

Master of Science Thesis

For obtaining the degree of Master of Science in Computational Science at University of Amsterdam

Anand Kumar Singh

Thursday 16th August, 2018

Student number:	11392045	
Thesis committee:	Dr. ir. Elia Bruni,	ILLC UvA, Supervisor
	Ir. Dieuwke Hupkes,	ILLC UvA, Supervisor
	Dr. ir. Rick Quax,	UvA, Examiner
	Prof. dr. Drona Kandhai,	UvA, Second Assessor
	Dr. ir. Willem Zuidema,	UvA, Third Assessor

An electronic version of this thesis is available at http://scriptiesonline.uba.uva.nl.

Informatics Institute · University of Amsterdam

UNIVERSITY OF AMSTERDAM

Copyright © Anand Kumar Singh
All rights reserved.

University of Amsterdam
Department Of
Computational Science

The undersigned hereby certify that they have read and recommend to the Department of Computational Science for acceptance a thesis titled **"Pondering in Artificial Neural Networks"** by **Anand Kumar Singh** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: Thursday 16th August, 2018

Supervisor:

Dr. ir. Elia Bruni

Supervisor:

Ir. Dieuwke Hupkes

Abstract

Abstract

Acknowledgements

Acknowledge

Amsterdam, The Netherlands
Thursday 16th August, 2018

Anand Kumar Singh

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Motivation

1.2 Objectives

Objectives

1.3 Outline

Thesis outline

Chapter 2

Background

This chapter provides a brief technical overview of the models that we will frequently encounter in the rest of the thesis. A fundamental understanding of these systems is essential for appreciating the problems associated with them and the potential solution for overcoming those problems, which will be discussed in the subsequent chapters.

2.1 RNN - A non linear dynamical system

We frequently encounter data that is temporal in nature. A few obvious examples would be, audio signals, videos signals, time series of a stock price and natural language (the holy grail of AI!). While traditional feed-forward neural networks are excellent at non linear curve fitting and classification tasks, it is unclear as to how they will approach the problem of predicting the value of a temporal signal T at time t given the states $T_0, T_1 .... T_{t-1}$ such that the states over time are not i.i.d. Human beings solve such problems by compressing and storing the previous states in memory and then using that information for predicting the state T_t. A conventional feed-forward network such as a MLP is inherently memoryless.

Recurrent neural networks overcome this restriction by having feedback loops which allow information to be carried from the current time step to the next. While the notion of implementing memory via feedback loops might seem daunting at first the architecture is refreshingly very simple. In a process known as unrolling a RNN can be seen as a sequence of MLPs (at different time steps) stacked together. More specifically, RNN maintains a memory across time-steps by projecting the information at any given time step t onto a hidden(latent) state through parameters θ which are shared across different time-steps. Fig 2.1 shows a rolled and unrolled recurrent network.

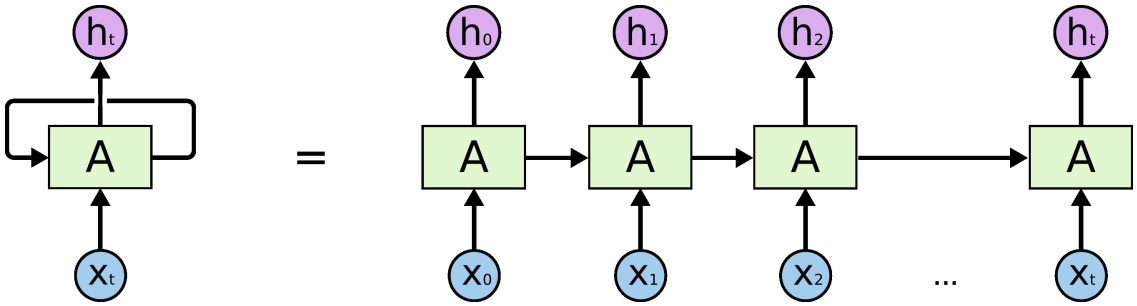


Figure 2.1: Schematic of a RNN Olah [2015]

2.1.1 Non linear dynamics inside a RNN cell

equations
presence of strange attractors.

$$h^{(t)} = f(x^{(t)}, h^{(t-1)}) \tag{2.1}$$

$$c_t = tanh(Ux_t + Wc_{t-1}) \tag{2.2}$$

$$y_t = sotmax(Vc_t) \tag{2.3}$$

2.2 BPTT and Vanishing Gradients

The optimization step i.e. the backward pass over the network weights is not just with regards to the parameters at the final time step but over all the time steps across which the weights (parameters) are shared. This is known as Back Propagation Through Time (BPTT) and it gives rise to the problem of vanishing (or exploding) gradients in vanilla RNNs. This concept is better elaborated upon through equation in the following section.

2.2.1 BPTT for Vanilla RNN

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\sum_t (y_t \log \hat{y}_t) \tag{2.4}$$

Since the weight W_{oh} is shared across all time steps, adding the derivatives across the sequence we get:

$$\frac{\partial \mathcal{L}}{\partial W_{oh}} = \sum_t \frac{\partial \mathcal{L}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial W_{oh}} \tag{2.5}$$

Now for time-step t $\rightarrow$ t+1:

$$\frac{\partial \mathcal{L}\,(t+1)}{\partial W_{hh}} = \frac{\partial \mathcal{L}\,(t+1)}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial W_{hh}} \tag{2.6}$$

Again since W_{hh} is shared across time steps, the gradient at t+1 will have contribution from the previous time steps as well. Summing over all the time steps we get:

$$\frac{\partial \mathcal{L}\,(t+1)}{\partial W_{hh}} = \sum_{\tau=1}^{t+1} \frac{\partial \mathcal{L}\,(t+1)}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_{\tau}} \frac{\partial \mathbf{h}_{\tau}}{\partial W_{hh}} \tag{2.7}$$

Aggregating over the whole sequence we get:

$$\frac{\partial \mathcal{L}}{\partial W_{hh}} = \sum_{t} \sum_{\tau=1}^{t+1} \frac{\partial \mathcal{L}\,(t+1)}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_{\tau}} \frac{\partial \mathbf{h}_{\tau}}{\partial W_{hh}} \tag{2.8}$$

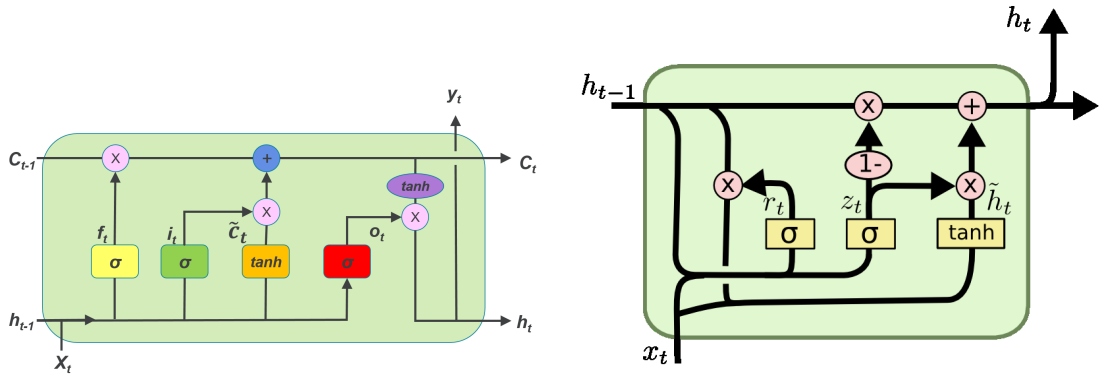
Looking at equation 2.8 we can see that the RNN gradient is a recursive product of $\frac{\partial h_t}{\partial h_{t-1}}$ and **in the event of this derivative being $\ll 1$ or $\gg 1$, the RNN gradient would vanish or explode respectively** when the network is trained over longer time-steps. In the former case the training would freeze while in the latter it would never converge. Therefore a vanilla RNN can't keep track of long term dependencies which is critical for tasks such as speech synthesis, music composition etc. The architectural modifications which solved the vanishing gradient problem and are the current de-facto RNN cell(s) are presented in the next section.

2.3 Gated RNNs

Looking at equation 2.1 we can see that RNN is an example of an iterated function and can therefore exhibit complex and chaotic behavior(substantiate this in later sections). However if for instance the RNN was to compute an identity function then the gradient computation wouldn't vanish or explode since the Jacobian is simply an identity matrix (rephrase this and substantiate with equation/previous equations). Now while an identity initialization of recurrent weights by itself isn't very interesting it brings us to the underlying principle behind gated architectures i.e. the mapping from memory state at one time step to the next is close to identity function.

2.3.1 LSTM

Long Short Term Memory (LSTM) introduced by Hochreiter and Schmidhuber [1997] is one of the two most widely used gated RNN architectures in use today. The fact that it has survived all the path-breaking innovations in the field of deep learning for over twenty years to still be the state of the art in sequence modeling speaks volumes about the architecture's ingenuity and strong fundamentals.



(a) LSTM **(b)** GRU

Figure 2.2: Gated RNNs

The fundamental principle behind the working of a LSTM is alter the memory vector only selectively between time steps such that the memory state is preserved over long distances (write this better). The architecture is explained as follows:

$$i = \sigma(x_t U^{(i)} m_{t-1} W^{(i)}) \tag{2.9}$$

$$f = \sigma(x_t U^{(f)} m_{t-1} W^{(f)}) \tag{2.10}$$

$$o = \sigma(x_t U^{(o)} m_{t-1} W^{(o)}) \tag{2.11}$$

$$\widetilde{c_t} = \tanh(x_t U^{(g)} m_{t-1} W^{(g)}) \tag{2.12}$$

$$c_t = c_{t-1} \odot f + \widetilde{c_t} \odot i \tag{2.13}$$

$$m_t = tanh(c_t) \odot o \tag{2.14}$$

The hidden state of a LSTM cell is the output which is then multiplied with the output weights followed by the softmax operation in order to yield the final output. The cell state is the memory of the LSTM unit. Furthermore the sigmoid activation for each one of the input, forget and output gates serves as a switch such that multiplying them element wise with other vector decides how much of the vector to filter out.

- **input modulation gate $g^{(t)}$:** The input modulation gate is computed based on the present input and the previous hidden state (which is exposed to the output). It yields a candidate memory for the cell state. Since it doesn't determine how much of a particular information to retain, it doesn't have the sigmoid activation. The tanh layer serves the purpose of creating candidate memories for the new cell state.

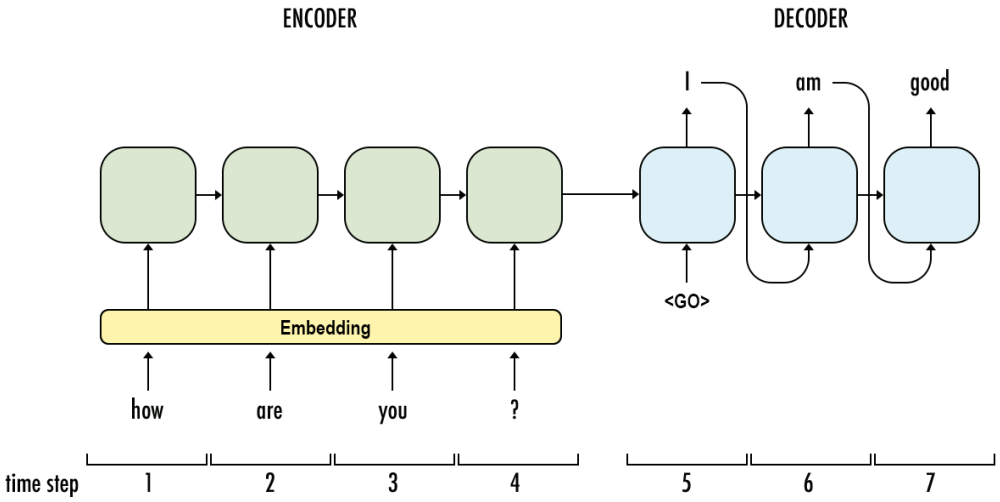


Figure 2.3: Schematic of a Seq2Seq ?

- **input gate** $i^{(t)}$**:** The input first computes a new cell state based on the new input and the previous hidden state and then decides how much of this information to "let through". A sum of the hadamard products of the (input gate,modulation gate) and (forget gate, previous cell state) tuples respectively yields the new cell state.
- **forget gate** $f^{(t)}$**:** The forget gate decides what to remember and what to forget for the new memory based on the current input and the previous hidden state.The sigmoid activation acts like a switch where 1 implies remember everything while 0 implies forget everything.
- **output gate** $o^{(t)}$**:** A tanh on the new cell state yields the candidate hidden state. The output gate then determines determines how much of this internal memory to expose to the top layers (and subsequent timesteps) of the network.

2.3.2 GRU

The GRU introduced by **?** area new type of gated RNN architecture which is has fewer parameters due to the lack of an output gate and is there computationally less complex in comparison to a LSTM.

Equations of GRU

2.4 Seq2Seq Models

Sequence-to-sequence (seq2seq) models introduced by Sutskever et al. , Cho et al. are a class of generative models that let us generate an output sequence when we are given an input sequence. They have found immense success in the fields of machine translation, speech recognition, question answering etc.

Details about the latent space, and the probabilistic generative model.

Chapter 3

Motivation

What I am actually looking at - Compositionality, Problems of RNN- SCAN, Lookup Tables. Why should it be compositional.

Pondering should be here as well.

Split it into background-technical and motivation-CH, Compositionality, Pondering etc.

3.1 Systematic Compositionality

3.2 Attention

3.2.1 Attention in Humans

Attention from a cognitive neuroscience perspective. Attentional blink

3.2.2 Se2Seq with Attention

It was shown by Cho et al. [2014] that the performance of a basic encoder-decoder models as explained in 2.4 is inversely related to the increase in length of the input sentence. Therefore in lines with concepts of the selective attention and attentional blink in human beings, Bahdanau et al. [2014] and later Luong et al. [2015] showed that soft selection from source states where most relevant information can be assumed to be concentrated during a particular translation step in NMT leads to improved performance.

Also cover this from a key-value pair pov.

3.3 Formal Language Theory

The field of formal language theory (FLT) concerns itself with the syntactic structure of natural language without much emphasis on the semantics. More precisely a formal language L is a sequence of strings with the constituent units/words/morphemes taken from a finite vocabulary Σ. It is more apt to define the concept of a formal grammar before proceeding further. A formal grammar G is a quadruple $< \Sigma, NT, S, R >$ where Σ is the vocabulary as previously defined, NT is the set of non-terminals, S the start symbol and R the set of rules. A rule can be expressed as $a \rightarrow b$ and can be understood as a substitution of a with b with a, b coming from Σ and/or NT. Now a formal language $L(G)$ can be defined as the set of all strings *generated* by grammar G such that the string consists of morphemes only from Σ, and has been generated by a finite set of rule (R) application after starting from S.

3.3.1 Chomsky Hierarchy

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3.3.2 Subregular Hierarchy

The simplest class of languages encountered in section 3.3.1 were regular languages that can be described using a FSA. If a language can be described by a mechanism even simpler than the FSA then it is a subregular language. While far from the expressive capabilities of regular languages which in turn are the least expressive class in the Chomsky hierarchy, subregular languages provide an excellent benchmark to test basic concept learning and pattern recognition ability of any intelligent system.

Strictly local languages We start with a string w and we are given a lookup table of k-adjacent characters known as *k-factors*, drawn from a particular language. The lookup table therefore serves the role of the language description. A language is k-local, if every k-factor seen by a *scanner* with a windows of size k sliding over the string w, is included in the aforementioned lookup-table.

Locally k-testable languages Instead of sliding a scanner over k-factors we consider all the k-factors to be atomic and build k-expression out of them using *propositional logic*. This language description is locally k-testable. As in the case of strictly local

languages, scanner won window size K slides over the string and records for every k-factor in vocabulary it's occurrence or nonoccurence in the string. The output of this scanner is then fed to a boolean network which verifies the k-expressions.

3.4 Pondering

Graves [2016]

Chapter 4

Datasets

4.1 Lookup Tables

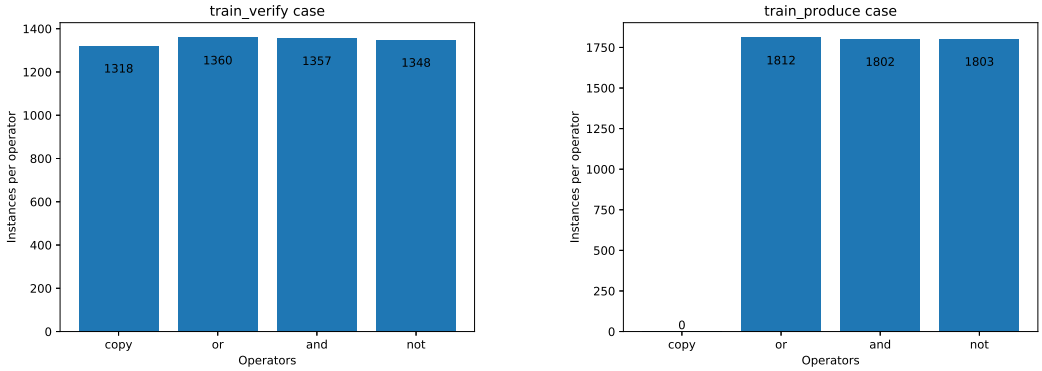
4.2 Symbol Rewriting

4.3 Micro Tasks

CommAI mini tasks [Baroni et al., 2017] are a set of strictly local and locally testable (please refer to section 3.3.2) languages designed to test the generalization capabilities of a compositional learner. Based on the mini tasks we propose a new dataset of sub-regular languages which we call he Micro Tasks. The language encompasses a vocabulary of all printable ascii characters, except printable but invisible characters i.e. tab, space, newline etc.

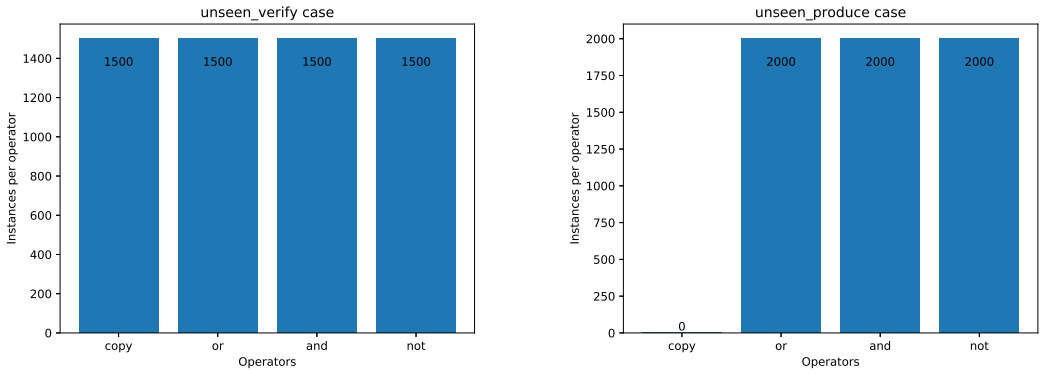
- Define verification and production tasks
- The vocabulary is acted upon by the boolean operators (or, and and not) in the case of both verification and production tasks. In case of verification we have an additional operation copy.
- Describe as in CommAI how copy and or are strictly local while and+not constitute locally k testable languages.

While the CommAI mini task position themselves as strictly local and locally testable tasks of progressive difficulty we add another layer of difficulty to the tasks by setting them up as either a decision problem(verify) or search problem(produce) in the same training domain.



(a) Verification Task **(b)** Production Task

Figure 4.1: Micro Tasks - Training Data



(a) Verification Task **(b)** Production Task

Figure 4.2: Micro Tasks - Unseen Data

Chapter 5

Proposed Models

The biggest criticism of deep learning is its overt dependence on voluminous data. For human level concept learning and generalization, deep learning models need to become more data efficient. I would like to break down the task of compositionality into two parallel sub-tasks as follows:

Storing Information meaningfully and efficiently

1. A more efficient data representation.
2. The new RNN cell designed with efficient memory allocation considerations.
3. Information Compression.

Retrieving Information efficiently

1. Attentive Guidance
2. Pondering
3. Curriculum Learning (Understander Executor).

Chapter 6

Experimental Setup

Maths. Merge with previous chapter

Chapter 7

Results and Discussions

Results

Chapter 8

Conclusions and Future Work

8.1 Conclusions

8.2 Recommendations

References

Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation : Encoder Decoder Approaches. *Ssst-2014*, pages 103–111, 2014. doi: 10.3115/v1/W14-4012.

Christopher Olah. Understanding LSTM Networks – colah's blog, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. 2015. ISSN 10495258. doi: 10.18653/v1/D15-1166. URL http://arxiv.org/abs/1508.04025.

Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. pages 1–19, 2016. ISSN 0927-7099. doi: 10.475/123. URL http://arxiv.org/abs/1603.08983.

Marco Baroni, Armand Joulin, Allan Jabri, Germàn Kruszewski, Angeliki Lazaridou, Klemen Simonic, and Tomas Mikolov. CommAI: Evaluating the first steps towards a useful general AI. pages 1–9, 2017. URL http://arxiv.org/abs/1701.08954.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. pages 1–15, 2014. ISSN 0147-006X. doi: 10.1146/annurev.neuro.26.041002.131047. URL http://arxiv.org/abs/1409.0473.

Sepp Hochreiter and Jurgen Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1–32, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735{%}5Cnfile:///Files/F