TOWARDS COMPOSITIONAL LEARNING FOR NATURAL LANGUAGE UNDERSTANDING

Arjun Verma

Master of Technology Thesis

June 2022

International Institute of Information Technology, Bangalore

TOWARDS COMPOSITIONAL LEARNING FOR NATURAL LANGUAGE UNDERSTANDING

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Technology

by

Arjun Verma
IMT2017008

International Institute of Information Technology, Bangalore
June 2022

This thesis is dedicated to the loving memory of my late grandfather

Thesis Certificate

This is to certify that the thesis titled **Towards Compositional Learning for Natural Language Understanding** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Arjun Verma**, **IMT2017008**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

_____

Gopalakrishnan Srinivasaraghavan

_____

Tim Klinger

Bangalore,

The 8th of June, 2022.

TOWARDS COMPOSITIONAL LEARNING FOR NATURAL LANGUAGE UNDERSTANDING

Abstract

Albeit their great success on a variety of tasks, modern deep learning architectures are plagued by their inability to capture abstract relationships efficiently. Learning a new concept requires these architectures to be trained on hundreds or thousands of training examples. Humans on the other hand possess the ability to understand concepts from very few examples. Many cognitive studies have prominently attributed this efficiency to our ability to exploit compositional structures. This has motivated the development of new specialized architectures with stronger compositional biases built into them. Unsurprisingly, one of the domains which has witnessed great research along this direction is of natural language understanding. The presence of well-defined compositional structures in the form of phrase structure grammars (grammar production rules) has benefitted this research greatly. In this thesis, we pursue a similar direction and introduce a novel compositional learning setup for natural language. We test our framework on the task of grammar induction and with promising initial results, we believe there is merit in adopting the proposed methodology.

Acknowledgements

Working on this thesis has not been an easy job at all. However, it could have been a lot worse had I not been lucky enough to have really great advisors in Prof. Srinivasaraghavan and Tim Klinger. I cannot thank them enough for the plethora of ideas and thoughtful discussions provided during the entire course of my thesis. I am also grateful to Matthew Riemer for having been an additional and truly commendable source of knowledge during these discussions.

Contents

List of Figures

List of Tables

List of Abbreviations

DL . . . . . Deep Learning

SOTA . . . . . State-Of-The-Art

CNN . . . . . Convolutional Neural Network

NLP . . . . . Natural Language Processing

PTB . . . . . Penn Treebank

LSTM . . . . . Long Short Term Memory

RIMs . . . . . Recurrent Independent Mechanisms

MoE . . . . . Mixture of Experts

IIITB . . . . . International Institute of Information Technology Bangalore

CHAPTER 1

INTRODUCTION

> "In terms of how much progress we've made in this field over the last two decades: I don't think we're anywhere close today to the level of intelligence of a two-year-old child. But maybe we have algorithms that are equivalent to lower animals for perception."
>
> (Yoshua Bengio, founder of Mila-Quebec AI Institute)

While the seed for modern deep learning's success was planted back in 1958 [2], only in the last decade has the emergent tree started to bear its fruits. Modern DL architectures have made significant accomplishments in domains such as object recognition, speech recognition and control. The advent of AlexNet [3] saw a complete domination of the architecture over the performances of any of the previous SOTAs on the popular ImageNet classification task. Since then, the field has never looked back with recent architectures such as DINO, SwinV2-G, YOLOR-D6 ([4], [5], [6]) continuing to perform well on even more challenging datasets. Hidden Markov Models had been the go-to framework to perform automatic speech recognition for a very long time [7]. However, the introduction of Alex Graves' work [8] on speech recognition with deep RNNs soon resulted in a huge paradigm shift with current SOTAs being completely end-to-end. The combination of deep learning with reinforcement learning has also worked wonders for solving complex control and planning problems. This has been

evident with the popular Alpha Zero [9] reaching superhuman level performance on games such as Shogi and Go.

With DL architectures making radical domain advancements, it becomes all the more important to ask whether there are any major inefficiencies that might lead to their demise. Leading connectionist pioneers such as Gary Marcus and François Chollet believe that **DL might be hitting a wall** [10]. This wall has its foundations grounded in the fact that current DL architectures are extremely data hungry and require large amounts of computational resources for efficient performance. Human beings, on the other hand, are capable of learning abstract relationships from very few samples of data. We are equipped with the ability to observe sparse amounts of data and learn its underlying causal mechanism. The knowledge obtained is then utilized to generalize far beyond the observed data. There is thus merit in understanding and trying to replicate human learning and cognition processes.

Cognitive scientists, developmental psychologists and linguists have often provided various windows into human cognition ([11], [12], [13]). The principle of *compositionality* is one of the key ideas to have originated from such studies. In its most naive sense, compositionality refers to the idea of assembling complex structures out of simpler parts. It is this principle that lies at the core of our remarkable ability to generalize and learn efficient abstract relations quickly ([14], [15]). In this thesis, we propose and test a novel learning framework that leverages compositionality for natural language understanding. We come up with a multi-purpose architecture capable of performing both, grammar induction and compositional representational learning simultaneously. However, for the scope of this thesis, we would only be focusing on testing the grammar induction capabilities of the architecture.

The rest of this thesis is organized as follows. In Chapter 2, we lay down a concrete motivation behind adopting the proposed framework. Chapter 3 focuses on some recent

advancements which have tried to leverage compositionality for their respective tasks. Chapter 4 deep dives into the description of the proposed framework and its learning setup. In Chapter 5, we report results of our conducted experiments to demonstrate that there is indeed merit in pursuing this research direction. Chapter 6 discusses the hindrances behind adopting the proposed framework in its current form. In the final chapter, we look into possible extensions to this research.

CHAPTER 2

MOTIVATION

2.1 Fundamental Problems with Deep Learning

> "For most problems where deep learning has enabled transformationally better solutions (vision, speech), we've entered diminishing returns territory in 2016-2017." (François Chollet, creator of Keras)

To understand the limitations of the current DL paradigm, it is first necessary to grasp what it actually is. DL is not a voodoo black-boxed end-all-be-all tool that it is often notoriously proclaimed to be. At its core, DL is nothing but a powerful statistical technique that is capable of recognizing patterns based on adequate sample data [16]. Given enough data, DL architectures are capable of learning any finite deterministic mapping between a set of inputs and paired outputs. In a utopian world of infinite data and infinite GPU capabilities, DL would probably have earned the sought-after title of 'The Master Algorithm' ([10], [17]). Sadly, such is not the case and we further discuss two critical problems that need to be solved for making significant advances further in the field.

2.1.1 Deep Learning is extremely data hungry

Humans possess the ability to learn new concepts from just a single example. DL on the other hand requires thousands of examples to "learn" the same concept with a similar adequacy [18]. Not only adults, even 7 month old infants have the capability to pick up abstract language-like rules from a small number of unlabelled examples [19]. Recent studies have also shown that human babies are born with innate biases to acquire language quickly [20]. As DL is mostly concerned with learning through superficial correlations, this huge necessity for data is not surprising.

However, this over-dependence on data is a far too crucial problem to be ignored. With DL finding its application in many critical scenarios such as the medical domain, the margin of error is often very low. To perform within these margins, the data requirements almost always shoot up exponentially. Gathering such quantities of data is generally infeasible and if doable, consumes large amounts of resources. Secondly, this data hungriness has also led to the following popular scenario : A big name such as Google or Facebook comes up with a modern breakthrough, but alas its replicability on a smaller scale is nearly impossible. This is due to the yin and yang nature of an architecture and its training data, and since these companies are the only ones with the data, the remarkable feat is often restricted to a publication with little applicability.

Geoffrey Hinton, one of the leading pioneers of DL, has also expressed significant concerns over this problem. Recognizing the human brain's ability to learn with very few examples, Hinton suggests getting rid of backpropagation, the very backbone of DL. He goes as far as to say, "My view is throw it all away and start again".

2.1.2 Deep Learning has no inherent bias to deal with hierarchies

Another remarkable drawback of the current DL paradigm is the ignorance of introducing human inspired architectural biases into models. For example, in natural language processing, most language models treat sentences as a mere sequence of words. On the other hand, linguists have long argued that natural language is hierarchical and have bestowed upon us several production rules to construct such hierarchies ([21] , [22]). Complex structures in language are often recursively constructed out of smaller primitive structures. The absence of an hierarchical bias in DL models leads to them failing to capture such recursive structures. Thus, they cannot capitalize on them by extending these structures to unseen scenarios. Contrastively, the ability to do so is generally known as *systematic compositionality* [23].

Marco Baroni and Brenden Lake have been the leading researchers to test systematic compositional skills of DL architectures. Their findings have demonstrated that modern DL architectures generalize well when the training and test set distributions don't differ much. However, when the distributions are intentionally diverged such that generalization now requires systematic compositional skills, these architectures fail spectacularly ([24], [25]). One can observe similar problems in the domains of planning and control as well. Very rarely have we seen reinforcement learning algorithms generalize their abstract plans in unseen scenarios. This is evident in the difficulty observed to achieve transfer learning across different Atari games [26].

Central to this problem is the fact that DL learns its correlations from a "flat" set of features [10]. Features are passed into models as a simple, non-hierarchical list where every element holds equal weightage. Thus, there seems to be an intrinsic architectural bias against an hierarchical learning setup. Current DL architectures such as transformers do try to capture these hierarchies via appended sequential positioning and learning relative relationships. However, these methods are still quite inadequate [25].

2.2 Compositionality

> "The world is compositional or God exists", since otherwise it would seem necessary for God to hard-wire human intelligence.
>
> (Stuart Geman, James Manning Professor of Applied Mathematics)

As stated previously, compositionality is the idea that novel complex representations can be created from a combination of basic primitive elements. It is a belief that the world is knowable and one can take things apart, understand them and mentally recompose them at will. The concept of compositionality is not at all new and has its very first mention by Gottfried Leibniz in 1666 [27]. It follows on the assumption that structures are composed hierarchically from elementary sub-structures utilizing a set of production rules. This encourages the fact that sub-structures and production rules may be learnt from finite amounts of data. They can then be used to generalize across different combinatorial scenarios.

Compositionality lies at the heart of productivity. It suggests that one can make an infinite number of representations from a finite number of primitives [28]. Compositional models would follow in the footsteps of the human mind, which can think of an infinite number of thoughts, construct infinite number of sentences, or learn new concept from an infinite space of possibilities ([29], [30], [31]). They would thus develop the skill to generalize out-of-distribution, to reason about and recognize scenarios utilizing the same underlying structures. Architectures implementing compositionality would also have the added benefits of being more interpretable. They would also possess the ability to generate quality synthetic data [32]. Inducing compositionality in DL architectures could thus play a major role in limiting the problems discussed earlier.

Ever since Baroni and Lake's seminal paper [23], there has been a lot of work on the compositional skills of neural networks ([33], [34], [35], [36], [37]). However,

one of the very first attempts on introducing compositionality in DL could be traced back to Hinton's Capsule Neural Networks (CapsNet) [38]. Capsule Neural Networks are meant to be an improvement over Convolutional Neural Networks (CNN). One of the biggest problems of a CNN is its inability to capture spatial relationships between objects. To achieve this, Hinton borrows concepts from computer graphics and suggests optimizing for equivariance instead of invariance. This results in capsules being able to capture part-whole relationships and being invariant to the pose of an object. Hinton recently rejuvenated this idea of capsules in a radically novel imaginary system named GLOM [39].

2.3 Compositionality in Natural Language

> "Colorless green ideas sleep furiously."
>
> (Noam Chomsky, Professor of Linguistics at MIT)

To quote Wilhelm von Humboldt, Chomsky's beloved thinker, "Language is the infinite employment of finite means." [40]. Natural language has the useful characteristic of being immensely productive. This means that it allows us to construct an infinite set of novel expressions. It is a widely accepted fact amongst linguists that compositionality promotes productivity [41]. In particular, the feat is commonly attributed to the principle of semantic compositionality, also called Frege's principle. According to the principle,

> *The meaning of a (syntactically complex) whole is a function only of the meanings of its (syntactic) parts together with the manner in which these parts were combined.* ([42])

This principle promotes the idea that the meaning of a complex expression is a

function of the meaning of its constituents and the rules governing their combination ([43], [44], [45]). It accounts for our ability to produce infinite expressions by dynamically recombining known components. Studies have given concrete evidence that if a system is not compositional, it will not be able to build complex semantic representations ([43], [24]). This is due to its inability to simultaneously combine syntactic and semantic elements.

Given their performance on machine translation tasks, there is no refuting the fact that modern NLP architectures are capable of generalizing beyond their training data [46]. What's interesting however is whether this performance depends solely on shallow heuristics or are these networks actually building grammar-based generalizations which would indicate an attempt to learn the skill of compositionality ([47], [48]). The authors of ([49], [50], [51]) have conducted extensive studies to demonstrate that recurrent networks are not simply relying on surface heuristics but are actually building pseudo-syntactic processing mechanisms. Their results show that the kind of productivity being exhibited by these networks possesses the ability to differentiate between grammatical and ungrammatical non-sensical sentences (refer to the quote at the beginning of this section [52] for a grammatical non-sensical sentence) that they weren't even fed during training. These studies provide clear evidence that successful language models are trying to mimic the skill of compositionality. Recent work by [53] has also hinted at transformer-based models' ability to achieve compositionality (although they require 112 million datapoints and 49 million parameters to do so). The argument thus goes that achieving compositionality is not the end goal. Rather, it is to come up with useful inductive biases that would help DL models to attain compositional skills in a data efficient manner. Such an ideology is very much in alignment with Chomsky's *poverty of stimulus* argument [54] and it is this direction that this thesis draws its inspiration from.

CHAPTER 3

METHODOLOGY

3.1 Dataset

We use Frege's principle as a guide to induce the notion of compositionality in our architecture. Following the principle would require us to combine the syntactic and semantic elements of natural language together. We turn to the Penn Treebank to gather our requirements for the syntactic elements.

3.1.1 The Penn Treebank

The Penn Treebank (PTB) [1], maintained by the University of Pennsylvania, is one of the largest human annotated dataset in NLP. It consists of approximately 7 million words of POS annotated text, 3 million words of skeletally parsed text, over 2 million words of text parsed for predicate-argument structure, and 1.6 million words of transcribed spoken text annotated for speech disfluencies [55]. We shall be utilizing a combination of the POS tagset and the syntactic tagset for our purposes. FC3.1 and FC3.2 provide a brief description of each of them.

We use the NLTK [56] package to work with PTB. As PTB is not a free corpus, we restrict ourselves to the 5% fragment of the corpus that NLTK provides. In pure

1. CC	Coordinating conjunction	25. TO	*to*
2. CD	Cardinal number	26. UH	Interjection
3. DT	Determiner	27. VB	Verb, base form
4. EX	Existential *there*	28. VBD	Verb, past tense
5. FW	Foreign word	29. VBG	Verb, gerund/present participle
6. IN	Preposition/subordinating conjunction	30. VBN	Verb, past participle
7. JJ	Adjective	31. VBP	Verb, non-3rd ps. sing. present
8. JJR	Adjective, comparative	32. VBZ	Verb, 3rd ps. sing. present
9. JJS	Adjective, superlative	33. WDT	*wh*-determiner
10. LS	List item marker	34. WP	*wh*-pronoun
11. MD	Modal	35. WP$	Possessive *wh*-pronoun
12. NN	Noun, singular or mass	36. WRB	*wh*-adverb
13. NNS	Noun, plural	37. #	Pound sign
14. NNP	Proper noun, singular	38. $	Dollar sign
15. NNPS	Proper noun, plural	39. .	Sentence-final punctuation
16. PDT	Predeterminer	40. ,	Comma
17. POS	Possessive ending	41. :	Colon, semi-colon
18. PRP	Personal pronoun	42. (	Left bracket character
19. PP$	Possessive pronoun	43.)	Right bracket character
20. RB	Adverb	44. "	Straight double quote
21. RBR	Adverb, comparative	45. ‘	Left open single quote
22. RBS	Adverb, superlative	46. “	Left open double quote
23. RP	Particle	47. ’	Right close single quote
24. SYM	Symbol (mathematical or scientific)	48. ”	Right close double quote

Figure FC3.1: The Penn Treebank POS tagset [1]

1.	ADJP	Adjective phrase
2.	ADVP	Adverb phrase
3.	NP	Noun phrase
4.	PP	Prepositional phrase
5.	S	Simple declarative clause
6.	SBAR	Clause introduced by subordinating conjunction or *0* (see below)
7.	SBARQ	Direct question introduced by *wh*-word or *wh*-phrase
8.	SINV	Declarative sentence with subject-aux inversion
9.	SQ	Subconstituent of SBARQ excluding *wh*-word or *wh*-phrase
10.	VP	Verb phrase
11.	WHADVP	*wh*-adverb phrase
12.	WHNP	*wh*-noun phrase
13.	WHPP	*wh*-prepositional phrase
14.	X	Constituent of unknown or uncertain category
	Null elements	
1.	*	“Understood” subject of infinitive or imperative
2.	0	Zero variant of *that* in subordinate clauses
3.	T	Trace—marks position where moved *wh*-constituent is interpreted
4.	NIL	Marks position where preposition is interpreted in pied-piping contexts

Figure FC3.2: The Penn Treebank syntactic tagset [1]

numbers, we are working with 3794 sentences out of the total 49,208 sentences that PTB contains. These sentences have been taken from around 2,499 stories from a three year Wall Street Journal (WSJ) collection.

3.1.2 Modifying the PTB

We don't directly use the parse trees from PTB for our purposes. Since we want our architecture to learn how to compose a given set of words, we need to feed it a list of transformation steps instead of the direct end product, the parse tree. We do this by utilizing NLTK to build a step by step transformation guide. Thus, given a sentence, our target is a list, where each element in the list is a tuple representing a step. Each tuple itself consists of two elements, where the second element denotes the span of words being combined and the first element denotes the non-terminal category of the combined span. The span of words to be combined is represented by a tensor of the same length as the input. If a word is a part of the span being merged at the current step, the tensor has a value of 1 at its index position. The rest of the values remain 0. The non-terminal category is a one-hot encoding built from the possible tagsets in FC3.1 and FC3.2. We maintain a right to left bottom-up convention while generating the transformation lists from parse trees. FC3.3 shows an example of the discussed modification procedure.

3.2 Architecture

In this section, we talk in detail about our proposed architectural setup. We first discuss the model we intend to use in our work, Recurrent Independent Mechanisms (or RIMs) [57]. We then elaborate on how RIMs can be adopted in a natural language setting to encourage compositionality.

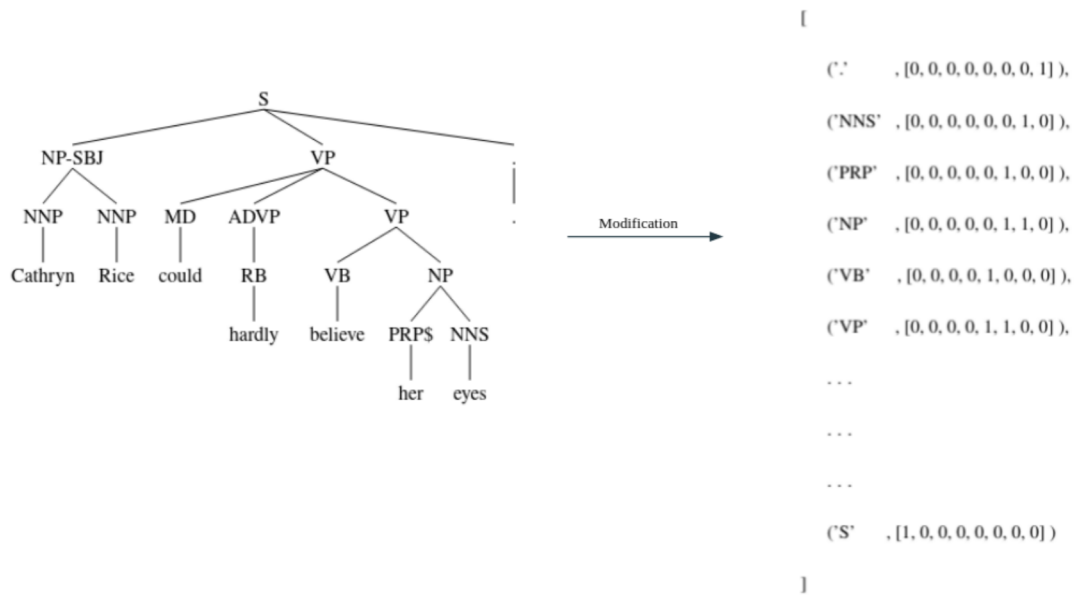

Figure FC3.3: Given the sample input : "Cathryn Rice could hardly believe her eyes.", we demonstrate the first six steps of the transformation list obtained after our modification procedure. As mentioned earlier, the parse tree on the left has been obtained from the PTB. Each tuple in the list represents a transformation step. The first element of a tuple is the abstraction category while the second element denotes the span of words being abstracted. Please note, for an ease of comprehension, we have left the abstraction category as is. In the actual dataset, they are represented as one-hot encodings. This transformation list serves as the target in our (input, target) pairs.

3.2.1 Recurrent Independent Mechanisms

RIMs were developed to utilize the fact that almost all physical processes in the world are modular and the various complexities are often just a combination of simpler sub-systems. A number of studies have concluded that even human cognition exploits this modular nature of environment ([58], [59]). Current DL models try to learn by recognizing regularities occurring in the world. Although these regularities emerge as statistical dependencies, they are ultimately produced by underlying dynamic processes which themselves are governed by various causal phenomena. These processes generally evolve independently and very rarely interact with each other. RIMs aim to take advantage of this phenomenon by trying to directly replicate the underlying mechanisms rather than uncovering statistical dependencies from data.

The way they do this is by designing an overall system which consists of independent sub-systems evolving over time. However, not all sub-systems evolve together at all times. At a time step, only the sub-systems displaying a significant overlap in their interactions are considered and evolved. Such models, where only a subset of mechanisms change and the rest remain invariant, could possibly learn to generalize well by better grasping the compositional structure of the world ([60], [61], [62]). The authors of RIMs show that incorporating such an architectural bias favoring modularity and dynamic recombination could prove to be quite beneficial when compared to their counterpart fully connected monolithic models [57].

3.2.2 Model

Before beginning the description of the model, we establish a convention for an ease of comprehension. From here on, the overall system would be referred to as **RIM Model**, whereas the independent sub-systems in the RIM Model would be called **RIM Units**.

One of the major architectural features of the RIM model is the use of attention mechanism [63] to isolate independent dynamics amongst RIM units. To refresh things, the attention mechanism is formulated as,

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d}})V,$$

where Q is the Query matrix, K is the Key matrix, V is the Value matrix and d is the dimension of each key

The reason behind this is that attention mechanism has the property of allowing neural networks to operate on sets of typed interchangeable arguments [57]. This property allows RIM units to act as "object-processing machines" capable of selecting only those parts of the input which are relevant to their functionality. To understand this, let's assume an input to be a set of objects. We now formulate an attention mechanism, where the keys and values are generated on each object in the input and the query is generated by a RIM unit. By doing so, we have now allowed the RIM units to process forward only those objects in the input which it would assign a high attention score to. Thus, in contrast to regular neural networks, where each neuron is capable of operating on only a fixed set of object (the outputs from the previous layer), each RIM unit can select which part of the input (objects) it would like to process forward. Since each RIM unit would produce its own query embedding, each unit is picking up objects relevant to its own mechanism thus ensuring independence amongst the varying RIM mechanisms [64].

With the core idea of the architecture in place, let us now go through a complete step of the RIM model when an input is passed into it.

3.2.2.1 Selecting active RIM units

As described earlier, we want a RIM unit to be activated and updated only when an input is relevant to it. The authors of [57] take care of this by formulating an attention based competition by which only a subset of RIM units get activated at each time step. Intuitively, RIM units compete at each time step to read from the input and only those RIM units that win this competition get to read and process their corresponding inputs. Let us now look at the selection procedure in a bit more detail.

The input x_t at time t is considered to be a set of elements. Each element is represented by a row in the matrix. The authors first concatenate a null vector to the end of this matrix. This can be shown as,

$$X = x_t \oplus \emptyset$$

Attention scores are now computed by generating queries from each RIM unit and keys, values from each element in the input. Thus,

$$K = XW^e, \text{one per input element and one for null vector}$$
$$V = XW^v, \text{one per input element and one for null vector}$$
$$Q = h_t W_k^q, \text{one per RIM unit}$$

where W^e is the weight matrix for generating keys, W^v is the weight matrix for generating values and W_k^q is the weight matrix for generating queries from k^{th} RIM unit's hidden state h_t at time t. The attention is thus computed as,

$$A_k^{(in)} = softmax(\frac{h_t W_k^q (XW^e)^T}{\sqrt{d_e}})XW^v,$$

where $\theta_k^{(in)} = (W_k^q, W^e, W^v)$ represents the parameters for k^{th} RIM unit and A^k are the attention scores computed over all the elements in X for the k^{th} RIM unit.

Now that the attention scores have been computed for all the RIM units, we select the top k_A RIM units to be activated. The top k_A units are the units putting the least attention score over the null vector (and thus attending the most to the actual input elements).

3.2.2.2 Evolving selected RIM units

After the selection mechanism has been applied and the appropriate RIM units have been activated, it is now time to evolve the independent dynamics of each unit. This seems like a good point to elaborate on what these RIM units actually are. In all its simplicity, a RIM unit is any recurrent neural network capable of capturing evolution of sequences over time. For our purposes, we take the RIM units to be LSTMs [65]. The independent dynamics are being captured in the evolution of the hidden states of the LSTMs.

To update the dynamics after a step, we need to compute new hidden states for each of the RIM units. Let the activated units be represented by the set S_t. For all units that are not activated, the hidden states remain unchanged, i.e,

$$h_{t+1,k} = h_{t,k}, \quad \forall k \notin S_t$$

It is important to note that the gradient still flows through an inactivated RIM unit.

For the RIM units which are activated, the hidden states evolve as,

$$h_{t+1,k} = LSTM(A_k^{(in)}, h_{t,k}; \quad \theta_k), \quad \forall k \in S_t$$

where $A_k^{(in)}$ represents the attended inputs for the k^{th} RIM and θ_k represents the parameters of the k^{th} RIM.

3.2.2.3 Communication between active RIM units

After the RIM units have transitioned independently, the authors of [57] allow for the RIM units to share information with each other. This is again done by utilising the attention mechanism. The idea goes that all activated RIM units are allowed to read from every other RIM unit (activated or not). The reason behind this being that since the inactivated units are irrelevant to the input at current time step, they need not be changed. However, the inactivated units may still contain some contextual information that might be relevant to the updation of the activated units. Taking the communication parameters for an activated RIM unit to be $\theta_k^{(C)} = (\tilde{W}_k^q, \tilde{W}^e, \tilde{W}^v)$, the updated hidden state for an activated unit after communication is thus given by,

$$\begin{aligned}
K_{t,k} &= \tilde{W}_k^e h_{t+1,k} \quad , \quad \forall k \\
V_{t,k} &= \tilde{W}_k^v h_{t+1,k} \quad , \quad \forall k \\
Q_{t,k} &= \tilde{W}_k^q h_{t+1,k} \quad , \quad \forall k \in S_t
\end{aligned}$$

$$h_{t+1,k} = softmax(\frac{Q_{t,k}(K_{t,:})^T}{\sqrt{d_e}})V_{t,:} \quad + \quad h_{t+1,k} \quad , \quad \forall k \in S_t$$

where $h_{t+1,k}$ is the final hidden state produced by the activated k^{th} RIM unit after an

input has been passed into the model.

3.2.3 Compositional Learning in Natural Language using RIMs

The authors of [57] tested the RIM model on a variety of tasks involving independent dynamics such as capturing the Newtonian motion of bouncing balls, Atari games and a few others. The results have been encouraging, showing that the RIM units successfully capture the dynamics involved in these tasks. We believe that the modular nature of the RIM model and its ability for dynamic recombination could be further exploited to promote compositional learning mechanisms. Cognitive scientists have already shown this natural connection between modularity and compositionality [66]. There have also been previous works involving the use of modular neural networks to improve systematic generalization ([67], [68]).

According to the best of our knowledge, no study has been conducted on inducing compositionality using RIMs. There have been attempts on utilizing RIMs for sentiment analysis and market volatility predictions ([69], [70]). However, both these works use the architecture as is and are widely different from our task of incorporating compositionality at hand. Broadly, we do this by trying to adapt the model to successfully induce grammar from sentences.

3.2.3.1 Interpretation

Before we begin with the architectural adaptation, we would like to specify why we think language is a natural fit for RIMs. We believe that the entire process of comprehension of a sentence could be viewed as a competition between multiple dynamic processes. Linguistic studies have shown that the language acquisition process in children is based on a ranking of heuristics phenomenon [71]. Children have an innate set of language heuristics that develop since infancy. These heuristics include a preference

for following the subject-object word order, a preference for assigning animate objects as subject, morphosyntactical cues, etc. When trying to comprehend a sentence, they try to maximize the number of heuristics that hold true. In case of conflicting heuristics, a natural ranking amongst them is preferred. Thus, one could assume that the RIM units would try to learn a bunch of similar abstract heuristics when fed in an input sentence.

3.2.3.2 RIMs for Grammar Induction

As briefly mentioned earlier, we try to achieve compositionality by training the RIM model to induce grammar. We do this by using the transformation lists we created during our dataset preparation procedure. Given an (input, target) pair, we begin by feeding in the input sentence at time t = 0. The input sentence is represented by a vector of word embeddings. We begin with the 100-dimensional GloVe embeddings [72] for our purposes.

Inside the RIM model, a null vector is first appended to the input sentence (as discussed in 3.2.2). After the appropriate RIM units have been selected, we turn towards the attention vectors being generated by these units. We interpret a weighted combination of the attention vectors as parts of the sentence the model focuses on at the current time step. The weights are generated as,

$$w_k = 0 \quad , \quad \forall k \notin S_t$$
$$w_k = softmax(1 - attn^k_{null\_vec}) \quad , \quad \forall k \in S_t$$

where, $attn^k_{null\_vec}$ indicates the attention score over the appended null vector by the k^{th} RIM unit. The weighted attention vector is thus given by,

$$attn_{combined} = \Sigma_k^R w_k \cdot attn_{1:n}^k$$

where, R indicates the total number of RIM units and $attn_{1:n}^k$ represents the attention scores for the k^{th} RIM unit over the n words in the sentence.

Ideally, if the model were to focus on spans of words in accordance to the sentence's parse tree, one would assume that the model has comprehended the sentence accurately. Recall that the transformation list helps us achieve exactly this. It contains a step-by-step guide to the creation of a sentence's parse tree. Following a right to left convention, each step provides us with a span of words to be abstracted into a non-terminal. Thus, by iterating over these transformations and generating a loss between the weighted attention vector and the target span at each time step t, the model could learn to build correct parse structures from sentences.

$$Loss^t = BinaryCrossEntropy(attn_{combined}^t, span_{transform}^t)$$

where, $span_{transform}^t$ is the ground truth span obtained from the step at time t of the transformation list of the input sentence (as discussed in 3.1.2). The reason for choosing binary cross-entropy as our loss function will be clarified in subsequent discussions.

We now focus on the hidden states being generated by the model. A weighted combination of the hidden states of the activated units is taken to be a representative of the non-terminal embedding generated at that time step. Using the same weights as before,

$$N.T_{embed} = \Sigma_k^R w_k \cdot h_{t+1}^k$$

where, R indicates the total number of RIM units and h_{t+1}^k represents the newly generated hidden state for k^{th} RIM unit.

This helps in defining our input sequence for the next time step. The sequence for the next time step is a vector of embeddings of the same dimensionality. For words which were not a part of the span being merged (as indicated by the step of the target transformation list), the embeddings in the vector remain unchanged. The generated non-terminal embedding is inserted in place of the span. The sequence is then appropriately padded with tensors of -1's to keep the dimensionality consistent across all time steps. This new sequence is passed into the model again for the next transformation step and the process repeats. For a particular sentence, the iterations continue until the entire transformation list is exhausted, i.e, the entire sentence has been replaced by the embedding for the non-terminal 'S'. Refer to FC3.4 and FC3.5 for our architectural setup.

We now discuss two key characteristics of our architecture :

- *Sigmoid "Attention" for span selection*

  One of the key changes that we make to [57] is the way the inputs are processed through the RIM units. While we keep the RIM unit selection mechanism the same, we don't pass the inputs weighted by their attention scores to the activated units. Instead, we replace the *softmax* function in the attention mechanism with a *sigmoid* function and use these as the weights for the inputs being passed to the active RIM units. To refresh, the input to the activated k^{th} unit now looks like,

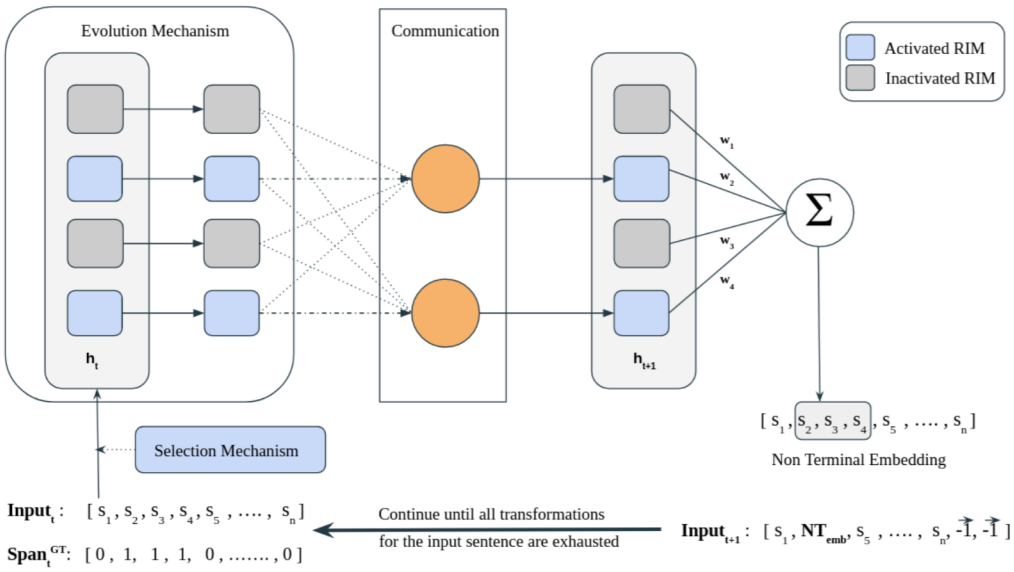

Figure FC3.4: Given an input sequence, it passes through the selection mechanism to activate the appropriate RIM units. The activated RIM units then undergo their respective evolution mechanisms. Once updated, the activated RIM units are now allowed to communicate with all other units in order to obtain a richer contextual information. Finally, a weighted combination of the hidden states is taken to be the embedding generated for the non-terminal at this transformation step. A copy of the input sequence is produced with the generated embedding replacing the span of words given by the ground truth ($Span_t^{GT}$). It is then appended by vectors of -1's to keep the dimensionality consistent with the input sequence. The new sequence is passed back into the model and the procedure repeats until all transformation steps for the sequence have been exhausted.

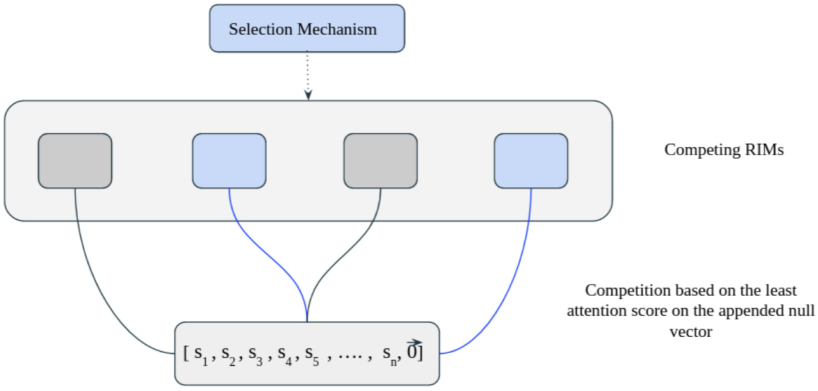

Figure FC3.5: The selection mechanism governs which RIM units will be activated based on the incoming sequence. This is done by enabling a 'reading competition' between the RIM units. The input sequence is first appended with a null vector. The top k_A units attending the least over this appended vector are then taken to be activated.

$$A_k^{(in)} = sigmoid(\frac{h_t W_k^q (XW^e)^T}{\sqrt{d_e}})XW^v$$

This has been done in a naive alignment with the viewpoint that grammar induction is a factored markovian process. Fitting this task into a markovian framework is not novel and has been done previously in ([73], [74]). This can be easily understood with the following example. A very popular phrase structure rule [75] in grammar is,

$$S \longrightarrow NP \quad VP$$

Given that a sentence just boils down to a noun phrase followed by a verb phrase, it does not matter how you reach this step. The entire history of parsing the sentence in a bottom-up fashion and reaching the stage of $\{NP \quad VP\}$ has been captured in this particular state. Thus, one can clearly see how the markovian property holds true for such a task. The factored side of the viewpoint comes from studies conducted by ([76], [77]). Word representations using different factors (such as POS tags or other word clusters) have shown to provide better context to language models and improve their performance. Recognizing grammar induction as a markovian process and representing words as factors, allows us to leverage the conditional independence property of factored-state MDPs [78]. Abstractly, this property ensures that the values of each state variable after a transition are determined independently of each other. These values are only conditioned on the previous state and action. For our purposes, we can interpret this property as treating the action on each word in a sentence independently of the actions on other words. This gives a natural argument to use sigmoid over softmax. Softmax, being a probability distribution over the inputs, does not allow conditional independence as it automatically lowers the relevance of other words having found a highly relevant one.

- *Guiding span selection in a Mixture of Experts fashion*

  Another key architectural change has been the loss generated on the weighted attention vector and the ground truth span. Contrastively, the authors of [57] train the model in a traditional manner of generating losses on the outputs at each step, i.e the updated hidden state vectors. We make this change in alignment with our earlier interpretation of the RIM units (as discussed in 3.2.3.1) where we treat each RIM unit as an independent heuristic trying to find spans of input relevant to its dynamics at each time step. The idea behind this is to let the RIM units take in whatever they think is relevant to them but they better combine well together to focus on the correct grammatical span. Cognitive studies have often suggested that the human brain acts as a "Mixture of Experts" (MoE). It has been said that different expert systems in the brain propose different strategies for action and the brain maintains mechanisms for switching behavioral control between them [79]. Taking a note from these cognitive mechanisms, we try to adopt a similar policy here. By keeping a track of the reliability of the predictions of each active RIM unit, our loss function maintains control in a manner that depends on the relative reliabilities across these units. Such MoE strategies have also been previously tried in [80], [81].

3.2.4 Expected Outcomes

By virtue of our above described adaptation, we are now in a position to test the following :

1. Given our learning setup, we can test whether the RIM model is able to reasonably generalize on the task of grammar induction.

2. If the above hypothesis holds, we potentially have the ability to make compositional embeddings for sentences. We can test the effectiveness of these embed-

dings on classification tasks such as sentiment analysis.

3. Again, if the first hypothesis holds, we can now abstract out sentences with each transformational step and can make compositional predictions on downstream tasks.

Within the scope of this thesis, we shall only be focusing on testing the grammar induction capabilities of the model.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Parser Evaluation

4.1.1 Standard Metric

The standard metric for evaluating a phrase structure parser is *bracket score*. According to this metric, the brackets found by the parser are compared to the brackets found in the gold standard parse tree. A bracket is analogous to a span of words to be merged together into a non-terminal. The usually reported scores of precision, recall and F1-score are as follows,

- Precision : Out of all brackets that the parser detects, how many are also present in the gold standard?
- Recall : Out of all brackets in the gold standard, how many does the parser also detect?
- F1-score : It is the harmonic mean between precision and recall,

$$\text{F1-score} = (2 * precision * recall)/(precision + recall)$$

However, our reported scores are not in accordance with these metrics. Since we

are working with only 5% of the PTB, our model is barely capable of being tested on such stringent metrics. To check whether our model is at least proceeding in the correct direction, we come up with a more lenient metric to evaluate its performance.

4.1.2 Our Metric

As previously mentioned in 3.2.3.2, the weighted attention vector ($attn_{combined}$) is considered to be a representative of the span of words that the model is currently trying to focus on. Ideally, we would want our attention scores to correspond significantly with the ground truth spans. This would serve as an as is procedure for bracketing. Our model could then be evaluated on aforementioned standard metrics. However, due to the small nature of our dataset, our attention scores don't converge and thus don't correspond significantly with the ground truth spans. Taking this into account, we decided to track a comparatively lenient metric.

Broadly, we consider our model to have done a good job if its identified the correct region to be focused upon at a particular transformation step. Thus, if our model has assigned the unique highest attention score within the indices representing the ground truth span, we take the prediction to be correct. Additionally, we also allow a relaxation window of size 2 on either sides of the span. Let us understand this concretely with the help of an example. Suppose, at a transformation step, words at indices 4 to 6 were to be combined into a non-terminal, we assume our model to have made an accurate prediction, if it assigns the unique maximum weight in $attn_{combined}$ at any of the indices between 2 to 8.

4.2 Experimental Setup

For our main results, we use a train-test split of 0.75-0.25 on the available corpus. After rigorous experimentation, we find the model initialized with 5 RIM units and top-3 selection mechanism to give the best results on our defined metric. The learning rate is controlled by Pytorch's ReduceLROnPlateau scheduler beginning with an initial rate of $2e-5$. We set the patience, threshold and factor parameters of the scheduler as 3, $1e-3$ and 0.5 respectively. These parameters imply that the learning rate will be reduced by half if the accuracy does not increase by at least $1e-3$ for 3 continuous epochs. We initialize the hidden states to all zero tensors and train the model for 11 epochs.

Before we report our results, we would like to raise two discussion points:

4.2.1 On Configurational Experimentation

We experimented with many variations of RIM units and top-k selection mechanism before settling on 5 and 3 as our parameters. We did not find any thumb-of-rule patterns to help us with our choices. This was expected as RIM units are supposed to be representatives of various heuristics observed in the language acquisition phenomenon. Extensive experimentation of individual units would be required to figure out which and what number of heuristics perform well together under different configurations. Having said that, one of the patterns that could certainly be expected, comes from keeping the top-k parameter as 1. Intuitively, allowing only one heuristic to take control of the sequence at a particular time step should almost always be inefficient. With an increasing number of RIM units and a constant top-1 selection mechanism, one would expect the performance to decrease as the number of competing representative heuristics increase. This expected pattern nearly holds true and has been captured in FC4.1.

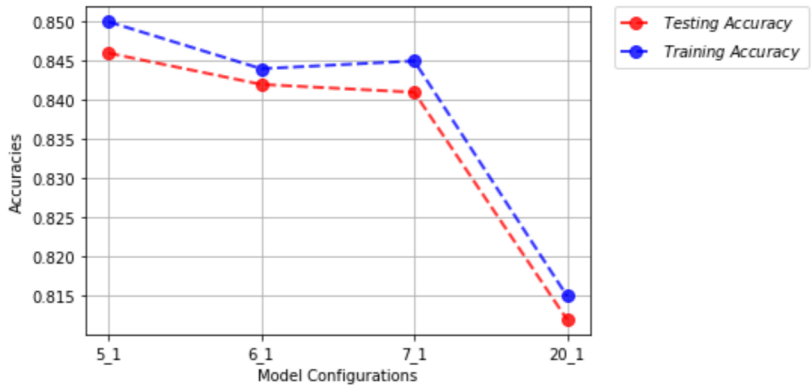

Figure FC4.1: The model configurations on the X-axis have been labelled as N_k, where N represents the number of RIM units in the model and k denotes the top-k selection mechanism. Y-axis captures the sentence-level accuracies on our previously defined metric. Sentence-level takes into consideration all possible non-terminals to be predicted.

4.2.2 On Visualizing our Outputs

It would certainly be useful to have a better understanding of our model predictions before diving into the main results. FC4.2 shows the plotted ground truth spans and weighted attention outputs at each transformation step (from Step 5 to 16) for a randomly selected sentence from the test set. One can observe how the model is trying to generate comparatively stronger attention scores in the vicinity of the words to be combined at a particular step. In an ideal scenario, we would expect the model generated attention scores to overlap comprehensively with the ground truth spans. However, as discussed previously, limited amounts of available data makes it difficult for the scores to converge identically with the ground truth. We thus make our evaluation metric quite lenient in comparison to the standard metrics.

4.3 Results

With a firm understanding of our model configuration and metrics, we are now ready to demonstrate our results.

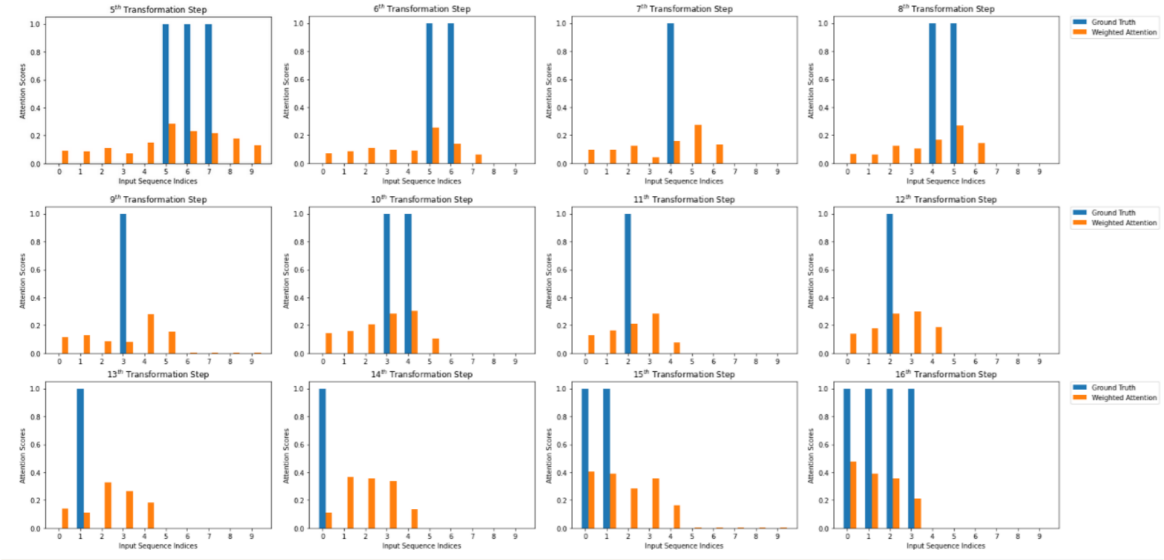

Figure FC4.2: A side-by-side comparison of weighted attention outputs and ground truth spans on the last 12 transformation steps (5 through 16) for a randomly selected sentence from test set.

4.3.1 Main Results

One of the most popular ways to generate parser metrics is by computing F1-scores on 6 major syntactic tags : NP, VP, PP, SBAR, ADJP and ADVP. We follow the same trend and generate our metrics on the aforementioned tags. We also compute the sentence-level accuracy by taking into consideration all possible syntactic tags to be predicted (which are 53 in number). These scores have been reported in TC4.1. We see that our model is doing a fairly good job when it comes to attending over spans being abstracted into these tags. In fact, in the next section we'll make an even broader statement by demonstrating that our model performs relatively well when more than one words are to be combined together.

Table TC4.1: Accuracies on individual syntactic tags

Model	NP	VP	PP	SBAR	ADJP	ADVP	Sentence
1-Layered (5, 3)	93.28	96.43	95.89	98.33	93.45	87.08	86.54

4.3.2 Case for allowed window

In TC4.4, we have divided all spans into two kinds : single-word and multi-word spans. As the name suggests, in single-word spans only one word gets abstracted into a non-terminal (e.g. 'name of a person' being abstracted into Noun (NN)) and more than one get abstracted in multi-word spans. Having divided into these categories, we now vary the window size from 0 to 2 and report the accuracies on each category. We observe that even with a window size of 0, where a prediction is considered correct only when the model assigns the unique highest attention score on one of the indices of the ground truth span, the multi-word span accuracies are already at 85 %. However, when one focuses on it's counterpart, it seems to be drastically underperforming with a 6 % accuracy. Increasing the window size to 1, significantly shoots up the accuracy for single-word spans, even more so if the window size is 2. These results indicate that our model is struggling to exactly overlap the attention scores with ground truth spans of single words. We believe that inadequate data is a significant cause behind this and put out a case for the same in next section.

Table TC4.2: Comparing single and multiple word abstraction accuracies by varying window sizes

Window Size	Single-Word Spans	Multiple-Word Spans
0	6	85
1	61	93
2	79	96

4.3.3 Deeper layers and case for more data

After tabulating the performance of our model, we try a deeper configuration in TC4.3. As the name suggests, in the 2-Layered model, we stack two RIM layers together and the input now passes through two layers of the same $(N,k) \sim (5,3)$ con-

figuration. The deeper model demonstrates an increased overall performance with the sentence-level metric being 87.23%. The accuracies on individual syntactic tags also show an upward trend.

Seeing promising improvements, we decided to train the deeper configuration for an increased number of epochs (20 instead of 11). This led to an overall sentence-level accuracy of 90.23%. The individual syntactic tag metrics show a minor to no improvement in their scores. However, we see interesting results when we track the single-word and multi-word span accuracies in TC4.4. The 2-Layered model trained until 11 epochs shows a significant increase over the 1-Layered model on single-spans. On training it further, we see a drop in performance over the single-word spans and an improvement for the multi-word spans. Thus, one can see that there is a clear tug of war between the accuracies on the two. We believe that with enough data, our model could eventually learn to perform better on both these categories and might even converge to exactly overlapping scores with the ground truth spans.

Table TC4.3: Comparing a deeper configuration

Model	NP	VP	PP	SBAR	ADJP	ADVP	Sentence
1-Layered (5, 3)	93.28	96.43	95.89	98.33	93.45	87.08	86.54
2-Layered (5, 3)	96.51	98.60	97.8	100.0	96.1	93.44	87.23

Table TC4.4: Comparing span scores (window size 0)

Category	1-Layered (11 epochs)	2-Layered (11 epochs)	2-Layered (20 epochs)
Single-Word	~ 6	~ 20	~ 15
Multi-Word	~ 89	~ 89	~ 91

CHAPTER 5

CONCLUSION

Making neural networks more compositional by means of structured modular architectures has been one of the central areas of research in AI [82]. There is a belief that such networks would make learning mechanisms more generalizable and efficient. In this thesis, we explore a similar direction and propose a new compositional learning setup by adapting RIMs for grammar induction. In its current form, due to the unavailability of sufficient data, we are unable to compare our setup against traditional benchmarks. However, as demonstrated in 4.3, our framework shows great promise and it would certainly be worthwhile to build upon this methodology.

Once the model has been improved to perform comparably to current benchmarks on grammar induction, there would be another line of interesting experiments to be conducted. These would involve testing the efficiency of the compositional sentence embeddings being generated by virtue of our training process. We hypothesize that these embeddings, enriched with both syntactic and semantic information, should be better than their counterpart traditional embeddings which are based only on semantic context. Simple experiments on downstream tasks such as sentiment analysis could be conducted to compare these embeddings. Additionally, our proposed multi-purpose framework also opens a window to make compositional predictions with each level of abstraction of the input sentence.

Linguistics is another domain that could benefit greatly from our framework. Studying the independent mechanisms being captured by the recurrent networks (LSTMs) in our model could help in better understanding the linguistic capabilities of neural networks. This could provide linguists an alternate view on the origin, scope and mechanisms underlying the language acquisition phenomena. Our framework could thus contribute in developing a more comprehensive account of the productivity and flexibility of natural language.

Bibliography

[1] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[2] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[4] Hao Zhang, Feng Li, Siyi Liu, Lei Zhang, Hang Su, Jun-Juan Zhu, Lionel M. Ni, and Heung yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *ArXiv*, abs/2203.03605, 2022.

[5] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution, 11 2021.

[6] Chien-Yao Wang, I-Hau Yeh, and Hong-yuan Liao. You only learn one representation: Unified network for multiple tasks, 05 2021.

[7] B. H. Juang and L. R. Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 38, 03 2013.

[9] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.

[10] Gary Marcus. Deep learning: A critical appraisal, 2018.

[11] Elizabeth S. Spelke. Principles of object perception. *Cognitive Science*, 14(1):29–56, 1990.

[12] Susan Johnson, Virginia Slaughter, and Susan Carey. Whose gaze will infants follow? the elicitation of gaze-following in 12-month-olds. *Developmental Science*, 1(2):233–238, 1998.

[13] Paul Bloom. How children learn the meanings of words. 2000.

[14] Brenden M. Lake, Tal Linzen, and Marco Baroni. Human few-shot learning of compositional instructions. *CoRR*, abs/1901.04587, 2019.

[15] Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. Learning with hierarchical-deep models. *IEEE transactions on pattern analysis and machine intelligence*, 35:1958–1971, 08 2013.

[16] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[17] Vasant Dhar and Pedro Domingos. Pedro domingos on the master algorithm : A conversation with vasant dhar. *Big Data*, 4:10–13, 03 2016.

[18] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[19] G.F. Marcus, S Vijayan, Shoba Bandi-Rao, and Peter Vishton. Rule learning by seven-month-old infants. *Science (New York, N.Y.)*, 283:77–80, 02 1999.

[20] Bálint Forgács, Tibor Tauzin, György Gergely, and Judit Gervain. The newborn brain is sensitive to the communicative function of language. *Scientific Reports*, 12(1):1220, Jan 2022.

[21] Noam Chomsky. A review of b. f. skinner's verbal behavior, 1959. This is one of the most influential articles in the history of the cognitive sciences. http://cogsci.umn.edu/millennium/final.html.

[22] Karl S. Lashley. The problem of serial order in behavior. 1951.

[23] Brenden Lake and Marco Baroni. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. 10 2017.

[24] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. 2017.

[25] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data, 2019.

[26] Girish Joshi and Girish Chowdhary. Cross-domain transfer in reinforcement learning using target apprentice, 2018.

[27] Gottfried Wilhelm Leibniz. *Dissertation on the Art of Combinations*, pages 73–84. Springer Netherlands, Dordrecht, 1989.

[28] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people, 2016.

[29] Jerry A. Fodor. *The Language of Thought*. Harvard University Press, 1975.

[30] Gary F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press, 2001.

[31] Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3–71, 1988.

[32] Liu Yuille. Limitations of deep learning for vision, and how we might fix them. *The Gradient*, 2019.

[33] Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. Compositional generalization by learning analytical expressions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11416–11427. Curran Associates, Inc., 2020.

[34] Chenyao Liu, Shengnan An, Zeqi Lin, Qian Liu, Bei Chen, Jian-Guang Lou, Lijie Wen, Nanning Zheng, and Dongmei Zhang. Learning algebraic recombination for compositional generalization, 2021.

[35] Juyong Kim, Pradeep Ravikumar, Joshua Ainslie, and Santiago Ontañón. Improving compositional generalization in classification tasks via structure annotations, 2021.

[36] Tim Klinger, Dhaval Adjodah, Vincent Marois, Josh Joseph, Matthew Riemer, Alex 'Sandy' Pentland, and Murray Campbell. A study of compositional generalization in neural models, 2020.

[37] Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. Compositional generalization for primitive substitutions. In *Proceedings of the 2019 Confer-*

ence on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4293–4302, Hong Kong, China, November 2019. Association for Computational Linguistics.

[38] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. NIPS'17, page 3859–3869, Red Hook, NY, USA, 2017. Curran Associates Inc.

[39] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network, 2021.

[40] Nigel Duffield. *v is for von Humboldt (discrete infinity)1*, page 270–303. Cambridge University Press, 2018.

[41] RICHARD MONTAGUE. Universal grammar. *Theoria*, 36(3):373–398, 1970.

[42] Francis Jeffry Pelletier. The principle of semantic compositionality. *Topoi*, 13(1):11–24, Mar 1994.

[43] Marco Baroni. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 375(1791):20190307, 2020.

[44] Zoltán Gendler Szabó. Compositionality. In *Stanford Encyclopedia of Philosophy*. 2008.

[45] Francis Pelletier. Context dependence and compositionality. *Mind Language - MIND LANG*, 18:148–161, 04 2003.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[47] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies, 2016.

[48] Shammur Absar Chowdhury and Roberto Zamparelli. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[49] Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 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)*, pages 1195–1205, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[50] Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. The emergence of number and syntax units in lstm language models, 2019.

[51] Tal Linzen and Marco Baroni. Syntactic structure from deep learning. *Annual Review of Linguistics*, 7(1):195–212, 2021.

[52] Noam Chomsky. *Syntactic structures*. The Hague: Mouton, 1957.

[53] Jacob Russin, Roland Fernandez, Hamid Palangi, Eric Rosen, Nebojsa Jojic, Paul Smolensky, and Jianfeng Gao. Compositional processing emerges in neural networks solving math problems. *CogSci ... Annual Conference of the Cognitive Science Society. Cognitive Science Society (U.S.). Conference*, 2021:1767–1773, Jul 2021. 34617074[pmid].

[54] Noam Chomsky. Rules and representations. *Behavioral and Brain Sciences*, 3(1):1–15, 1980.

[55] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: An overview. 01 2003.

[56] Michael Elhadad. Natural language processing with python steven bird, ewan klein, and edward loper (university of melbourne, university of edinburgh, and bbn technologies) sebastopol, ca: O'reilly media, 2009, xx+482 pp; paperbound, isbn 978-0-596-51649-9, $44.99; on-line free of charge at nltk.org/book. *Computational Linguistics*, 36:767–771, 12 2010.

[57] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms, 2019.

[58] Danielle S. Bassett and Michael S. Gazzaniga. Understanding complexity in the human brain. *Trends in cognitive sciences*, 15(5):200–209, May 2011. 21497128[pmid].

[59] Peter Carruthers. Is the mind a system of modules shaped by natural selection? In Christopher R. Hitchcock, editor, *Contemporary Debates in the Philosophy of Science*. Blackwell, 2003.

[60] Yoshua Bengio. The consciousness prior, 2017.

[61] Herbert A. Simon. The architecture of complexity. 1991.

[62] Jonas Peters, Dominik Janzing, and Bernhard Schlkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

[63] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[64] Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. Learning independent causal mechanisms. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine*

Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 4036–4044. PMLR, 10–15 Jul 2018.

[65] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[66] Bob van Tiel, Mark Blokpoel, Iris van Rooij, and Andrea E Martin. Compositionality, modularity, and the architecture of the language faculty. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 43, 2021.

[67] Davide Locatelli. *Measuring Compositional Generalization in Modular Neural Networks*. PhD thesis, Imperial College London, 2020.

[68] Laura Ruis and Brenden Lake. Improving systematic generalization through modularity and augmentation, 2022.

[69] Parsa Bagherzadeh and S. Bergler. Multi-input recurrent independent mechanisms for leveraging knowledge sources: Case studies on sentiment analysis and health text mining. In *DEELIO*, 2021.

[70] Jie Yuan and Zhu Zhang. Market volatility prediction based on long- and short-term memory retrieval architectures. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20, New York, NY, USA, 2020. Association for Computing Machinery.

[71] Petra Hendriks. The acquisition of compositional meaning. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 375(1791):20190312, 2020.

[72] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[73] Walt Woods. Rl-grit: Reinforcement learning for grammar inference, 2021.

[74] Edward C. Williams, Nakul Gopalan, Mine Rhee, and Stefanie Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 1–7. IEEE Press, 2018.

[75] Noam Chomsky. *Aspects of the theory of syntax*. MIT, Cambridge, MA, 1965. Synopsis of the standard theory. The second chapter "is concerned with the base of the syntactic component, and with difficulties that arise from the assumption that it is, strictly speaking, a phrase structure grammar" (preface, p. vi).

[76] Philipp Koehn and Hieu Hoang. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[77] Jan Niehues, Thanh-Le Ha, Eunah Cho, and Alexander H. Waibel. Using factored word representation in neural network language models. In *WMT*, 2016.

[78] Alexander L. Strehl, Carlos Diuk, and Michael L. Littman. Efficient structure learning in factored-state mdps. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, page 645–650. AAAI Press, 2007.

[79] John O'Doherty, Sangwan Lee, Reza Tadayonnejad, Jeffrey Cockburn, Kiyohito Iigaya, and Caroline J Charpentier. Why and how the brain weights contributions from a mixture of experts, Jun 2020.

[80] Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. Moebert: from bert to mixture-of-experts via importance-guided adaptation, 2022.

[81] Phong Le, Marc Dymetman, and Jean-Michel Renders. Lstm-based mixture-of-experts for knowledge-aware dialogues, 2016.

[82] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering, 2016.