



THÈSE DE DOCTORAT

DE L'UNIVERSITÉ PSL

Préparée à l'Ecole normale supérieure

Dans le cadre d'une cotutelle avec l'université de Tel Aviv

**Modélisation du langage par réseau de neurones,
approche par simplicité**

**Simplicity-Based Language Modeling Using
Artificial Neural Networks**

Soutenue par
Nur LAN
Le 7 mai 2024

Ecole doctorale n° 158
**Cerveau, Cognition,
Comportement**

Spécialité
Sciences cognitives

Composition du jury :

Giorgio MAGRI Directeur de recherche CNRS, Paris 8	<i>Président</i>
Hava SIEGELMANN Provost Professor, University of Massachusetts Amherst	<i>Rapporteuse</i>
Noga ZASLAVSKY Assistant professor, University of California, Irvine	<i>Examinatrice</i>
Roni KATZIR Associate professor, Tel Aviv University	<i>Co-encadrant de thèse</i>
Emmanuel CHEMLA Directeur de recherche CNRS, Ecole normale supérieure	<i>Directeur de thèse</i>

Contents

Acknowledgments	3
Introduction	6
1 Minimum Description Length Recurrent Neural Networks (joint with Michal Geyer, Emmanuel Chemla, and Roni Katzir)	10
2 Benchmarking Neural Network Generalization for Grammar Induction (joint with Emmanuel Chemla and Roni Katzir)	26
3 Bridging the Empirical-Theoretical Gap in Neural Network Formal Language Learning Using Minimum Description Length (joint with Emmanuel Chemla and Roni Katzir)	37
4 Large Language Models and the Argument From the Poverty of the Stimulus (joint with Emmanuel Chemla and Roni Katzir)	50
Résumé français	113

Acknowledgments

This work is due to two brilliant minds who I was very fortunate to have as advisors – Roni Katzir and Emmanuel Chemla. I will thank them in chronological order of meeting.

Roni's seminar about learning, computation, and cognition at TAU was a revelation both literally and figuratively: coming in I did not know what any of it meant, and coming out I knew that I wanted to have a part in it. Roni's teaching holds you captivated from beginning to end, traversing all fields of science and often touching on the philosophical and the sublime. Yet it always ends in concrete research questions that combine all of these together in the most original and elegant way. I learned from him how to think critically, how to strive for meaningful research questions, and above all how to distinguish competence from performance, not only in linguistics. I am thankful to him for always being available, both for listening and for sharing his infinite knowledge with endless kindness and patience, and for explaining things again and again until I would finally understand.

I first met Emmanuel in late 2019 when I arrived at ENS as a visiting student. At that time I was simply hoping to join his research on animal linguistics which I liked from afar. Fortunately for me this meeting led to four magical years of learning from Emmanuel about topics as diverse as neural networks, signaling games, semantics, human and animal cognition, and so much more. This turned out to be a common experience: one usually starts working with Emmanuel on a single project only to discover that he also knows about, and can do, literally everything else. On top of that, somehow, Emmanuel is also the kindest and most thoughtful advisor, colleague, and friend one can hope to have. This combination made my work so much easier than it should have been. I am grateful to him for always being there to listen to my nonsense ideas, and for what I believe is his biggest talent: to abstract away from earthly thoughts and turn them into non-nonsense ones.

I owe my studies at ENS to Philippe Schlenker, who initially made my visit there possible, and was later present in them through enlightening meetings and lectures, and through his never-ending line of works that I keep revisiting for inspiration. I am grateful to Paul Egré and Benjamin

Spector for their insightful guidance as members of my thesis committee. And I thank the jury of my thesis defense – Hava Siegelmann, Giorgio Magri, and Noga Zaslavsky – for their valuable feedback on this work and during the defense. I especially thank Hava for sharing her knowledge in person and for inspiring parts of this thesis through her seminal works.

Most everything I know about neural networks is due to Yair Lakretz and his illuminating course on biological and artificial networks which he originally taught at TAU. It was a pleasant coincidence to end up in the same office as Yair in Paris a few years later, thanks to which I had free access to a constant source of knowledge about the brain, academia, science, and philosophy.

Meeting Amir Anvari was the rare kind of occasion where one recognizes greatness in real time. Any conversation with Amir, whether about linguistics, computer science, politics or movies, stays with you for a lifetime. I deeply envy his ability to read thoughts and to put them back together for you in the most concise and poetic way. I thank him deeply for doing this often enough in my presence, which helped me understand what it was that I was actually trying to do.

I thank Mathias Sablé-Meyer for his kind support and for inspiring me at different points of my studies, both through stimulating discussions and by always being much ahead with his original and beautiful research.

While at ENS I was lucky to be in the presence of some very smart people who are fortunately also very nice. I thank Nadine Bade, Jeanne Bruneau--Bongard, Keny Chatain, Cécile Crimon, Megan Dailey, Pablo Diego, Émile Enguehard, Diego Feinmann, Carlo Geraci, Maria Giavazzi, Michael Goodale, Nicolas Guerin, Janek Guerrini, Jeremy Kuhn, Sharon Peperkamp, Bowei Shao, and Chloé Tahar for making ENS such a pleasant and stimulating place to work in throughout those years.

At TAU I was equally fortunate to meet and work with talented and lovely people: Matan Abudy, Adi Behar-Medrano, Itamar Shefi, and Imry Ziv. I especially thank Ezer Rasin who was present in this journey from the start and showed me that it's possible to do meaningful and rigorous work while also being extremely nice and generous. Working with Iddo Berger was my initiation in elegant and clean programming, and I still try to live up to his standards. My passion for most of the topics explored in this work was born thanks to Yishay Mansour and his fantastic

course on computational models and complexity.

I am indebted to Danny Fox for making my fascinating visit to MIT possible and for eye-opening discussions. During this visit I was also fortunate to meet some brilliant people who I thank for their generous input on this work and beyond it: Bob Berwick, Martin Hackl, and Roger Levy. I also thank Ido Benbaji, Omri Doron, Janek Guerrini, Adèle Hénot-Mortier, Pelin Kivrak, Keely New, and Anastasia Tsilia who made my time in Cambridge pleasant.

I was fortunate to meet Dominique Sportiche for prosaic matters which led to an acquaintance with not only an exceptionally kind person but also a very unique linguistic mind. I thank him for multiple conversations spanning linguistics, French syntax, and life itself, and for making my first years in Paris such a magical time.

Avner Shavit originally inspired this journey by passing on to me two passions of his: living in Paris and doing a PhD there. He is also the funniest person I know, which helped calm my nerves during difficult times.

Claire Daum is to thank for large parts of this thesis, for generously opening her home, known as Bambalère, which allowed me to work surrounded by green fields and jumping deer.

I am grateful to the following furry and funny friends who helped me avoid anthropocentrism: Boten, Putzi, Belle, Bubbs, and Schwartz.

I thank my friends Lior Bosak, Nir Neumann, and Aviaz Rand who kept me visiting Israel despite all circumstances. Above all I thank my mom Irit and my brother Ynon, for their love from afar during all this time. And I am grateful for a beautiful flower named Yasmine who brightens my days and keeps me smiling.

Introduction

“...the function of the brain and nervous system and sense organs is in the main eliminative and not productive. [... Its function is] to protect us from being overwhelmed and confused by this mass of largely useless and irrelevant knowledge, by shutting out most of what we should otherwise perceive or remember at any moment, and leaving only that very small and special selection which is likely to be practically useful. [...] To make biological survival possible, Mind at Large has to be funneled through the reducing valve of the brain and nervous system. What comes out at the other end is a measly trickle of the kind of consciousness which will help us to stay alive on the surface of this Particular planet. To formulate and express the contents of this reduced awareness, man has invented and endlessly elaborated those symbol-systems and implicit philosophies which we call languages.”

— Aldous Huxley, *The Doors of Perception*, 1954

Human learning is idiosyncratic in Nature, with learning outcomes ranging from acquisition of language to mathematical and musical reasoning and the discovery of scientific laws. All sufficiently complex animals learn, yet human learning seems unique with respect to the mental representations that are used when assembling hypotheses about the world. One leading explanation that cuts across such highly varied tasks postulates that human mental representations take distinct properties such as discreteness and recursiveness.

Beyond the formal properties of learned knowledge, another open question relates to the learning process: how might a learning system reach such results based on a restricted amount of data. Here, one explanation that has been offered for this problem assumes a general cognitive pressure for simplicity, which drives both generalization and accuracy. This pressure can operate both at an intentional, conscious level (‘Occam’s razor’, for example in the scientific method) as well at an unconscious, automatic level (e.g., during language acquisition). It is unknown whether

such a pressure is shared with other species.

The two elements above are necessary components of any learning system: a representational space and an objective function. Alongside them, a learning system also needs to implement a search algorithm – a strategy for navigating the representational space of all possible learning outcomes, guided by the objective function. Little is still known about the implementation of any of these three components in humans, let alone in other species. Arriving at a full specification of all three is a hard problem that currently involves efforts across multiple fields such as cognitive science, neuroscience, and linguistics. The task is already hard enough in humans, despite the advantage of introspection which facilitates the question of mental representation to some extent. Probes in other species, whose mental representations remain extremely opaque, are even harder.

The task might be easier in artificial models. The ability to freely manipulate any of the components above (the representational space, the objective function, and the search algorithm) makes it possible to hypothesize about their implementation elsewhere, both in isolation and as a whole, and to compare outcomes of different configurations with human data.

In this thesis we will mainly explore the questions of the objective function and of the representational space, using the currently most widely used model – artificial neural networks (ANNs). Despite reaching ubiquity across many domains, ANNs still struggle on a range of tasks that require human-level generalization. Moreover, their inner workings are opaque, and inspecting their acquired knowledge, including the formal properties of their inner representations, remains a difficult task. The goal of the thesis then is to shed light on these shortcomings and to discover how one would approach fixing them in ANNs, in a way that might also be informative of human cognition.

Chapter 1 is a proposal of a new type of recurrent neural networks (RNNs), in which standard training objective functions are replaced/complemented with an objective to minimize the Minimum Description Length (MDL) of the network itself. Applying this principle, one obtains much smaller RNNs, by design, and also RNNs which can learn, in full generality, some formal languages that were outside the reach of standard ANNs.

Chapters 2 and 3 then suggest that a shift in objective function might be required in order to

reach human-level generalization in ANNs. In Chapter 2 we propose a systematic way to inspect the generalization capabilities of artificial models using formal languages. We release several benchmarking datasets that unify and standardize previous ambiguous and inconclusive results in the literature. The performance of existing ANNs, even ones equipped with more expressive architectures, is shown to underperform compared to the MDL model from Chapter 1. In Chapter 3 we manually build a network which perfectly captures the language $a^n b^n$, and show that it does not lie at optima of standard objectives. Instead, this perfect network does lie at an optimum point in terms of an MDL objective.

In Chapter 4 we set aside the question of the training objective and probe the linguistic knowledge acquired by large language models (LLMs). Here we do this from the perspective of the Argument From the Poverty of the Stimulus (APS) – a long-debated claim for an innate capacity for language in humans, given the alleged scarcity of linguistic input available to a child. We show that contrary to previous claims, LLMs struggle to acquire satisfactory knowledge of syntactic phenomena for which humans have clear-cut judgements (Parasitic Gaps and Across-the-Board movement). Given that modern LLMs are trained on massive amounts of data, which go many orders of magnitude beyond the linguistic experience of humans, we conclude that these failures support the claim that humans are endowed in specific ways that make it possible to acquire such phenomena from much less input, i.e., supporting the APS.

Much more remains to be done. On a technical level, the MDL objective currently poses a hard optimization problem, which leaves results limited to small-scale tasks such as formal language learning (which as we show, are nevertheless challenging even for state-of-the-art models). More efficient exploration of the MDL search space, in the form of better hardware or better search algorithms, would unleash new possibilities for tasks such as natural language learning. Regarding the representational space of ANNs, our results suggest that ANNs lack core properties which push human learners in the direction of highly formalistic outcomes such as Parasitic Gaps and Across-the-Board movement. Building such restrictive properties into ANNs remains a difficult task, given ANNs' convoluted inner workings on the one hand, and their powerful expressivity on the other.

In terms of implications for cognition, human or other, our results strengthen existing proposals that a pressure for simplicity plays an important role in learning, at least in humans; and that human linguistic competence relies on representations with particular formal properties. Yet much more remains to be determined, such as the interplay between the objective function and the two other components of the learning system (representational space and search algorithm). The different configurations of the system generate a wide range of predictions: it is possible, for example, that a simplicity pressure is equally at work in other species, but that it operates over a completely different representational space, currently impenetrable to us. Similarly, committing to a certain representational space and an objective function, one could then ask what search algorithm needs to be implemented by the brain in order to reach the unique learning outcomes we see in humans.

Chapter 1

Minimum Description Length Recurrent Neural Networks (joint with Michal Geyer, Emmanuel Chemla, and Roni Katzir)

Nur Lan, Michal Geyer, Emmanuel Chemla, Roni Katzir; Minimum Description Length Recurrent Neural Networks. *Transactions of the Association for Computational Linguistics* 2022; 10: 785–799. doi: https://doi.org/10.1162/tacl_a_00489

Minimum Description Length Recurrent Neural Networks

Nur Lan^{1,2}, Michal Geyer², Emmanuel Chemla^{1,3,*}, Roni Katzir^{2,*}

¹Ecole Normale Supérieure

²Tel Aviv University

³EHESS, PSL University, CNRS

{nlan, chemla}@ens.fr

michalgeyer@mail.tau.ac.il

rkatzir@tauex.tau.ac.il

Abstract

We train neural networks to optimize a Minimum Description Length score, i.e., to balance between the complexity of the network and its accuracy at a task. We show that networks optimizing this objective function master tasks involving memory challenges and go beyond context-free languages. These learners master languages such as $a^n b^n$, $a^n b^n c^n$, $a^n b^{2n}$, $a^n b^m c^{n+m}$, and they perform addition. Moreover, they often do so with 100% accuracy. The networks are small, and their inner workings are transparent. We thus provide formal proofs that their perfect accuracy holds not only on a given test set, but for any input sequence. To our knowledge, no other connectionist model has been shown to capture the underlying grammars for these languages in full generality.

1 Introduction

A successful learning system is one that makes appropriate generalizations. For example, after seeing the sequence 1,0,1,0,1 we might suspect that the next element will be 0. If we then see 0, we might be even more confident that the next input element will be 1. Artificial neural networks have shown impressive results across a wide range of domains, including linguistic data, computer vision, and many more. They excel at generalizing when large training corpora and computing resources are available, but they face serious challenges that become particularly clear when generalizing from small input sequences like the one presented above.

First, they tend to overfit the learning data. To avoid this, they require external measures to control their own tendency for memorization (such as regularization) as well as very large training

corpora. Moreover, standard regularization techniques fall short in many cases, as we show below.

Second, even when successful, they tend to produce non-categorical results. That is, they output very high probabilities to target responses, but never 100%. Adequate, human-like generalization, on the other hand involves having both a probabilistic guess (which neural networks can do) and, at least in some cases, a clear statement of a categorical best guess (which neural networks cannot do).

Third, these networks are often very big, and it is generally very hard to inspect a given network and determine what it is that it actually knows (though see [Lakretz et al., 2019](#) for a recent successful attempt to probe this knowledge in the context of linguistics).

Some of the challenges above arise from the reliance of common connectionist approaches on backpropagation as a training method, which keeps the neural architecture itself constant throughout the search. The chosen architecture must therefore be large enough to capture the given task, and it is natural to overshoot in terms of size. Furthermore, it must allow for differentiable operations to be applied, which prevents certain categorical patterns from even being expressible.

In this paper, we propose to investigate a training method which differs from common approaches in that its goal is to optimize a Minimum Description Length objective function (MDL; [Rissanen, 1978](#)). This amounts to minimizing error as usual, while at the same time trying to minimize the size of the network (a similar pressure to a Bayesian size prior). As a result, the objective function offers a guide to determining the size of the network (a guide that error minimization alone does not provide), which means that the architecture itself can evolve during learning and typically can decrease in size. One potential side effect is that optimization cannot be done through back-

*Both authors contributed equally to this work.

propagation alone. We here use a genetic algorithm to search through the very large search space of neural networks of varying sizes.

We find that MDL-optimized networks reach adequate generalizations from very small corpora, and they avoid overfitting. The MDL-optimized networks are all small and transparent; in fact, we provide proofs of accuracy that amount to infinite and exhaustive test sets. They can also provide deterministic outputs when relevant (expressing pure 100% confidence). We illustrate this across a range of tasks involving counters, stacks, and simple functions such as addition.

2 Previous work

Our primary concern in this paper is the objective function. The idea of applying a simplicity criterion to artificial neural networks dates back at least to [Hinton and Van Camp \(1993\)](#), who minimize the encoding length of a network’s weights alongside its error, and to [Zhang and Mühlenbein \(1993, 1995\)](#), who use a simplicity metric that is essentially the same as the MDL metric that we use in the present work (and describe below). [Schmidhuber \(1997\)](#) presents an algorithm for discovering networks that optimize a simplicity metric that is closely related to MDL. Simplicity criteria have been used in a range of works on neural networks, including recent contributions (e.g., [Ahmadizar et al., 2015](#) and [Gaier and Ha, 2019](#)). Outside of neural networks, MDL — and the closely related Bayesian approach to induction — have been used in a wide range of models of linguistic phenomena, in which one is often required to generalize from very limited data (see [Hornung, 1969](#), [Berwick, 1982](#), [Stolcke, 1994](#), [Grünwald, 1996](#), and [de Marcken, 1996](#), among others, and see [Rasin and Katzir, 2016](#) and [Rasin et al., 2021](#) for recent proposals to learn full phonological grammars using MDL within two different representational frameworks). In the domain of program induction, [Yang and Piantadosi \(2022\)](#) have recently used a Bayesian learner equipped with a simplicity prior to learn formal languages similar to the ones we present below.

Turning to the optimization algorithm that we use to search for the MDL-optimal network, our work connects with the literature on using evolutionary programming to evolve neural networks. Early work that uses genetic algorithms for various aspects of neural network optimization in-

cludes [Miller et al. \(1989\)](#), [Montana and Davis \(1989\)](#), [Whitley et al. \(1990\)](#), and [Zhang and Mühlenbein \(1993, 1995\)](#). These works focus on feed-forward architectures, but [Angeline et al. \(1994\)](#) present an evolutionary algorithm that discovers recurrent neural networks and test it on a range of sequential tasks that are very relevant to the goals of the current paper. Evolutionary programming for neural networks remains an active area of research (see [Schmidhuber, 2015](#) and [Gaier and Ha, 2019](#), among others, for relevant references).

Our paper connects also with the literature on using recurrent neural networks for grammar induction and on the interpretation of such networks in terms of symbolic knowledge (often formal-language theoretic objects). These challenges were already taken up by early work on recurrent neural networks (see [Giles et al., 1990](#) and [Elman, 1990](#), among others), and they remain the focus of recent work (see, e.g., [Wang et al., 2018](#) and [Weiss et al., 2018a](#)). [Jacobsson \(2005\)](#) and [Wang et al. \(2018\)](#) provide discussion and further references.

3 Learner

3.1 Objective: Minimum Description Length

Consider a hypothesis space \mathcal{G} of possible grammars, and a corpus of input data D . In our case, \mathcal{G} is the set of all possible network architectures expressible using our representations, and D is a set of input sequences. For a given $G \in \mathcal{G}$ we may consider the ways in which we can encode the data D given G . The MDL principle ([Rissanen, 1978](#)), a computable approximation of Kolmogorov Complexity ([Solomonoff, 1964](#); [Kolmogorov, 1965](#); [Chaitin, 1966](#)), aims at the G that minimizes $|G| + |D : G|$, where $|G|$ is the size of G and $|D : G|$ is the length of the shortest encoding of D given G (with both components typically measured in bits). Minimizing $|G|$ favors small, general grammars that often fit the data poorly. Minimizing $|D : G|$ favors large, overly specific grammars that overfit the data. By minimizing the sum, MDL aims at an intermediate level of generalization: reasonably small grammars that fit the data reasonably well.

The term $|D : G|$ corresponds to the surprisal of the data D according to the probability distribution defined by G (i.e., the negative log of the probability assigned to targets by the network). The term $|G|$ depends on an encoding scheme for mapping networks onto binary strings, described below.

3.2 Our networks and their encoding

The MDL learner explores a space of directed graphs, made of an arbitrary number of units and weighted connections between them. We describe the actual search space explored in the experiments below, by explaining how these networks are uniquely encoded to produce an encoding length $|G|$.

3.2.1 Example

We will consider networks such as the one represented in Fig. 1. It consists of two input units (yellow units 1 and 2) and one output unit with a sigmoid activation (blue unit 3). The network has one forward connection (from unit 1 to 3) and one recurrent connection (unit 2 to 3), represented by a dashed arrow. Recurrent connections feed a unit with the value of another unit at the previous time step, and thus allow for the development of memory across the different time steps of the sequential tasks we are interested in. Here, unit 3 is fed with input 2 from the previous step. The connection weights are $w_{1,3} = 0.5$ and $w_{2,3} = 2$. Unit 3 is also fed a bias term $b_3 = 1$ represented by a sourceless arrow.

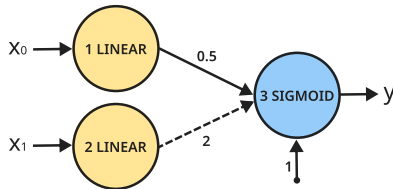


Figure 1: Example network, encoded in Fig. 2

We will now explain how such a network is represented to measure its encoding size $|G|$.

3.2.2 Preliminary: encoding numbers

To ensure unique readability of a network from its string representation we use the prefix-free encoding for integers from Li and Vitányi (2008):

$$E(n) = \underbrace{11111\dots1111}_{\text{Unary enc. of } \lceil \log_2 n \rceil} \underbrace{0}_{\text{Separator}} \underbrace{10101\dots00110}_{\text{Binary enc. of } n}$$

3.2.3 Encoding a network

The encoding of a network is the concatenation of (i) its total number of units, and (ii) the ordered concatenation of the encoding of each of its units.

3.2.4 Units

The encoding of a unit includes its activation function, the number of its outgoing connections, the

encoding of each of its outgoing connections, and its bias weight, if any.

3.2.5 Activation functions

Possible activation functions are: the linear activation (identity), ReLU, sigmoid, square, as well as the floor function and the unit step function (returns 0 for inputs ≤ 0 and 1 otherwise). To build an intuitive measure of simplicity into the model’s choice of activation functions, we add a cost to each function, encoded as a unary string: the linear activation has no cost; square costs 2 bits; ReLU, sigmoid, and floor cost 4 bits; and the unit step function costs 8 bits.

3.2.6 Connections and weights

A connection’s encoding includes its target unit number (each connection is specified within the description of its source unit, hence the source needs not be encoded), its weight, and its type: forward (0) or recurrent (1).

To simplify the representation of weights in classical neural networks and to make it easier to mutate them in the genetic algorithm described below, we represent weights as signed fractions $\pm \frac{n}{d}$, which are serialized into bits by concatenating the codes for the sign (1 for +, 0 for -), the numerator and the denominator. For example, the weight $w_{ij} = +\frac{2}{5}$ would be encoded as:

$$\underbrace{1}_{+} \underbrace{E(2) = 10\dots10}_2 \underbrace{E(5) = 1110\dots11}_5$$

w_{ij}

3.3 Search algorithm

Our interest in this paper is the MDL objective function and not the training method. However, identifying the MDL-optimal network is hard: the space of possible networks is much too big for an exhaustive search, even in very simple cases. We therefore need to combine the objective function with a suitable search procedure. We chose to use a genetic algorithm (GA; Holland, 1975), which frees us from the constraints coming from backpropagation and is able to optimize the network structure itself rather than just the weights of a fixed architecture. For simplicity and to highlight the utility of the MDL metric as a standalone objective, we use a vanilla implementation of GA, summarized in Algorithm 1.

The algorithm is initialized by creating a population of N random neural networks. A network is

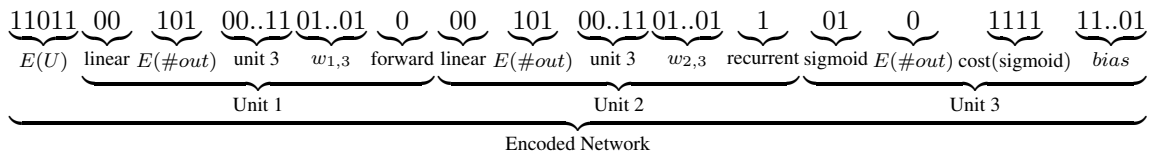


Figure 2: Binary encoding of the network in Fig. 1

Algorithm 1 Genetic algorithm

function TOURNAMENTSELECTION(pop):

$T \leftarrow t$ random networks from pop

$winner \leftarrow argmin_{MDL}(T)$

$loser \leftarrow argmax_{MDL}(T)$

return $winner, loser$

end function

$population \leftarrow \emptyset$ \triangleright Population initialization

while $|population| < N$ **do**:

generate a random network net

add net to $population$

end while

$generation \leftarrow 0$ \triangleright Evolution loop

while $generation < Gen$ **do**:

for N steps **do**:

$parent, loser \leftarrow$

TOURNAMENTSELECTION($population$)

$offspring \leftarrow mutate(parent)$

$eval(offspring)$ \triangleright MDL score

remove $loser$ from $population$

add $offspring$ to $population$

end for

$generation \leftarrow generation + 1$

end while

return $argmin_{MDL}(population)$

initialized by randomizing the following parameters: activation functions, the set of forward and recurrent connections, and the weights of each connection. Networks start with no hidden units. In order to avoid an initial population that contains mostly degenerate (specifically, disconnected) networks, output units are forced to have at least one incoming connection from an input unit.

The algorithm is run for Gen generations, where each generation involves N steps of selection followed by mutation. During selection, networks compete for survival on to the next generation based on their fitness, i.e., their MDL score, where lower MDL is better. A selected network

is then mutated using one of the following operations: add/remove a unit; add/remove a forward or recurrent connection; add/remove a bias; mutate a weight or bias by changing its numerator or denominator, or flipping its sign; and change an activation function. These mutations make it possible to grow networks and prune them when necessary, and to potentially reach any architecture that can be expressed using our building blocks. The mutation implementations are based on Stanley and Miikkulainen (2002).¹

On top of the basic GA we use the Island Model (Gordon and Whitley, 1993; Adamidis, 1994; Cantú-Paz, 1998) which divides a larger population into ‘islands’ of equal size N , each running its own GA as described above, periodically exchanging a fixed number of individuals through a ‘migration’ step. This compartmentalization serves to mitigate against premature convergence which often occurs in large populations. The simulation ends when all islands complete Gen generations, and the best network from all islands is taken as the solution.

4 Experiments

We ran tasks based on several classical formal-language learning challenges. We use both deterministic and probabilistic expectations to test the ability of a learner to work on probabilistic and symbolic-like predictions. In addition to showing that the MDL learner performs well on test sets, we provide proofs that it performs well on the whole infinite language under consideration.

¹A mutation step can potentially produce a network that contains loops in its non-recurrent connections, most commonly after a new connection is added. In the feed-forward phase, we detect loop-closing connections (using depth-first search) and ignore them. This avoids circular feeding, and at the same time creates a smoother search space, in which architectures freely evolve, even through intermediate defective networks. Stagnant loop connections which don’t end up evolving into beneficial structures are eventually selected out due to the $|G|$ term.

4.1 General setup and comparison RNNs

All simulations reported in this paper used the following hyper-parameters: 250 islands, each with population size 500 (total 125,000 networks), 25,000 generations, tournament size 2, migration size 2, and a migration interval of 30 minutes or 1,000 generations (earliest of the two). The number of generations was chosen empirically to allow enough time for convergence. Each task was run three times with different random seeds.²

To compare the performance of the MDL-optimized recurrent neural networks (MDLRNNs) with classical models, we trained standard RNNs on the same tasks, varying their architecture — GRU (Cho et al., 2014), LSTM (Hochreiter and Schmidhuber, 1997), and Elman networks (Elman, 1990) — as well as the size of their hidden state vectors (2, 4, 32, 128), weight regularization method (L1/L2/none), and the regularization constant in case regularization was applied ($\lambda = 1.0/0.1/0.01$), totaling 84 RNN configurations. Each configuration was run three times with different random seeds. These RNNs were trained with a cross-entropy loss, which corresponds to the $|D : G|$ term divided by the number of characters in the data.³

Table 1 summarizes the results for both MDL-RNNs and classical RNNs for all the tasks that will be described below. For each task, the representative network for each model (out of all configurations and random seeds) was chosen based on performance on the test set, using MDL scores for MDLRNNs and cross-entropy for RNNs.

It should be noted that model selection based on test performance is at odds with the premise of MDL: by balancing generalization and data fit during training, MDL automatically delivers a model which generalizes well beyond the training data; MDL also does away with the post-hoc, trial-and-error selection of hyper-parameters and regularization techniques inherent in classical models. In other words, MDL models can just as well be se-

²All experimental material and the source code for the model are available at <https://github.com/taucompling/mdlrnn>.

³All RNNs were trained using the Adam optimizer (Kingma and Ba, 2015) with learning rate 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The networks were trained by feeding the full batch of training data for 1,000 epochs. The cross-entropy loss for RNNs is calculated using the natural logarithm, converted in Table 1 to base 2 for comparison with MDL scores.

lected based on training performance. This is in contrast to standard RNNs, for which the training-best model is often the one that simply overfits the data the most. We show that even when given post-hoc advantage, RNNs still underperform.^{4,5}

4.2 General measures

We will report on two measures: (i) **Accuracy**, which we explain below based on each task’s unique properties; (ii) **Cross-entropy**, averaged by character for better comparability across set sizes, and compared with a baseline value calculated from the probability distribution underlying each task. Most originally, in some occasions we report on these measures on whole, infinite languages. This is possible because, by design, MDL-optimized networks are just large enough for their task, allowing us to fully understand their inner workings.

4.3 Experiment I: counters and functions

We test our model’s ability to learn formal languages that require the use of one or multiple counters: $a^n b^n$, $a^n b^n c^n$, $a^n b^n c^n d^n$. These languages can be recognized using unbounded counting mechanisms keeping track of the number n of a ’s and balance the other characters accordingly. We also test our model’s ability to learn languages which not only encode integers as above, but also operate over them: $a^n b^m c^{n+m}$ (addition) and $a^n b^{2n}$ (multiplication by two). We did not test general multiplication through the language $a^n b^m c^{nm}$ for practical reasons, namely that the length of the sequences quickly explodes.

4.3.1 Language modeling

The learner is fed with the elements from a sequence, one input after the next, and at each time

⁴When models are selected based on training performance (and then evaluated on the test sets), MDLRNNs outperform standard RNNs in all tasks in terms of cross-entropy and accuracy. We make the full training-based comparison available as part of the experimental material.

⁵Training-based selection yields different MDLRNN winners for three out of the seven relevant tasks when trained on the smaller data sets, and for two tasks when trained on the larger sets. However, only one of these cases, for $a^n b^n c^n$ with the larger training set, results in a drop from 100% accuracy when selected by test to a suboptimum (97.6%), while other models remain at the same accuracy levels. MDL optimization is thus not immune to overfitting, which could occur for example due to accidental bad sampling. However, as made visible by our results, MDL training produces models that generalize well across data sets.

	Training set size	Test cross-entropy ($\times 10^{-2}$)			Test accuracy (%)		Best RNN		MDLRNN proof
		MDLRNN	RNN	optimal	MDLRNN	RNN	Type	Size	
$a^n b^n$	100	29.4	53.2	25.8	100.0	99.8	Elman	2	Th. 4.1
	500	25.8	51.0	25.8	100.0	99.8	Elman	2	
$a^n b^n c^n$	100	49.3	62.6	17.2	96.5	99.8	Elman	4	Th. 4.2
	500	17.2	55.4	17.2	100.0	99.8	Elman	4	
$a^n b^n c^n d^n$	100	65.3	68.1	12.9	68.6	99.8	GRU	4	
	500	13.5	63.6	12.9	99.9	99.8	GRU	4	
$a^n b^{2n}$	100	17.2	38.0	17.2	100.0	99.9	Elman	4	Th. 4.3
	500	17.2	34.7	17.2	100.0	99.9	GRU	4	
$a^n b^m c^{n+m}$	100	39.8	47.6	26.9	98.9	98.9	Elman + L1	128	Th. 4.4
	500	26.8	45.1	26.9	100.0	98.9	Elman	128	
Dyck-1	100	110.7	94.5	88.2	69.9	10.9	Elman	4	Th. 4.5
	500	88.7	93.0	88.2	100.0	10.8	LSTM	4	
Dyck-2	20,000	1.19	1.19	1.18	99.3	89.0	GRU	128	
Addition	100	0.0	75.8	0.0	100.0	74.9	Elman	4	Th. 4.6
	400	0.0	72.1	0.0	100.0	79.4	Elman	4	

Table 1: Performance of the networks found by the MDL model compared with classical RNNs for the tasks in this paper. Test accuracy indicates *deterministic accuracy*, the accuracy restricted to deterministic steps; Dyck-n tasks have no deterministic steps, hence here we report *categorical accuracy*, defined as the fraction of steps where a network assigns a probability lower than $\epsilon = 0.005$ to each of the illegal symbols. When available, the last column refers to an infinite accuracy theorem for MDL networks: describing their behavior not only for a finite test set but over the relevant, infinite language.

step its task is to output a probability distribution for the next character. Following Gers and Schmidhuber (2001), each string starts with the symbol #, and the same symbol serves as the target prediction for the last character in each string.

If the vocabulary contains n letters, the inputs and outputs are one-hot encoded over n input units (in yellow in the figures), and the outputs are given in n units (in blue). To interpret these n outputs as a probability distribution we zero negative values and normalize the rest to sum to 1. In case of a degenerate network that outputs all 0's, the probabilities are set to the uniform value $1/n$.

4.3.2 Setup

Each task was run with data set sizes of 100 and 500. The training sets were generated by sampling positive values for n (and m , when relevant) from a geometric distribution with $p = .3$. The maximal values K observed for n and m in our batches of size 100 and 500 were 14 and 22, respectively.

We test the resulting networks on all unseen sequences for n in $[K+1, K+1001]$. For $a^n b^m c^{n+m}$ we test on n and m in $[K+1, K+51]$, i.e., the subsequence 2,500 unseen pairs.

Only parts of the sequences that belong to the formal languages presented here can be predicted deterministically, e.g., for $a^n b^n$, the deterministic parts are the first a (assuming $n > 0$), all b 's ex-

cept the first one, and the end-of-sequence symbol. For each of the tasks in this section, then, we report a metric of **deterministic accuracy**, calculated as the number of matches between the output symbol predicted with maximum probability and the ground truth, relative to the number of steps in the data that can be predicted deterministically.

4.3.3 Results

The performance of the resulting networks is presented in Table 1. In Figures 3-6, we show the networks that were found and their typical behavior on language sequences. Thanks to their low number of units and connections, we are able to provide simple walkthroughs of how each network operates. We report the following measures:

Deterministic accuracy: perfect for almost all tasks, both with small and large training sets.

The MDL learner achieves perfect accuracy for the tasks $a^n b^n$ and $a^n b^{2n}$, both with small and large training sets. The learner also achieves perfect accuracy for $a^n b^n c^n$ and $a^n b^m c^{n+m}$ with a larger training set, and in fact the networks found there would be better optima also for the respective smaller training sets, therefore showing that the suboptimal results for the small training sets are only due to a limitation of the search, and that perfect accuracy should in principle be reachable there too with a more robust search.

The only task for which MDLRNNs did not reach 100% accuracy is $a^n b^n c^n d^n$. Since the other tasks show that our representations make it possible to evolve counters, we attribute this failure to the search component, assuming a larger population or more generations are needed, rather than lack of expressive power; networks for this task require more units for inputs and outputs, which enlarge the number of possible mutations the search can apply at each step.

Cross-entropy: near perfect. For all tasks but $a^n b^n c^n d^n$, the MDLRNN per-character average cross-entropy is also almost perfect with respect to the optimal cross-entropy calculated from the underlying probability distribution.

RNNs: no perfect generalization. Among the competing models, no standard RNN reached 100% deterministic accuracy on the test sets, and all RNNs reached suboptimal cross-entropy scores, indicating that they failed to induce the grammars and probability distributions underlying the tasks. In terms of architecture size, the best-performing RNNs are often those with fewer units, while L1 and L2 regularizations do not yield winning models except for one task.

Transparency supports formal proofs that results are perfect for the whole, infinite language. For all tasks but $a^n b^n c^n d^n$ then, deterministic accuracy and cross-entropy are perfect/excellent on training and test sets. Because the MDL networks are small and transparent, we can go beyond these results and demonstrate formally that the task is performed perfectly on the entire infinite underlying language. To our knowledge, such results have never been provided for any classic neural network in these tasks or any other.

Theorem 4.1. *The $a^n b^n$ network represented in Fig. 3 outputs the correct probability for each character, for each sequence in the $a^n b^n$ language, with a margin of error below 10^{-6} .*

Proof. Table 2 traces the value of each unit at each step in a legal sequence for the relevant network. When normalizing the outputs to obtain probabilities, the values obtained are the exact ground-truth values, up to the contribution of $\sigma(-15)$ to that normalization (sigmoid is abbreviated as σ), which is negligible compared to all other positive values (the largest deviance is $\frac{\sigma(-15)}{1+\sigma(-15)} \approx 3 \cdot 10^{-7}$,

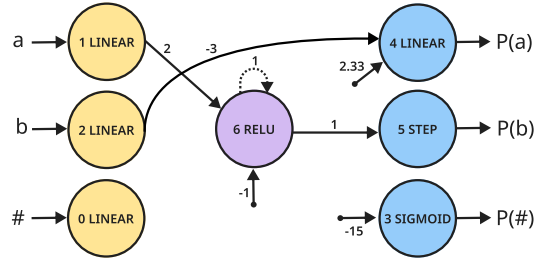


Figure 3: The network found by the MDL learner for the $a^n b^n$ task, for a training set with data set size 500. See Theorem 4.1 for a description of how this network accepts any $a^n b^n$ sequence and why it rejects any other sequence.

$a^n b^n$	Unit 6	Unit 4 $P(a)$	Unit 5 $P(b)$	Unit 3 $P(\#)$
Initial #	0	$7/3$	0	$\sigma(-15)$
k^{th} a	k	$7/3$	1	$\sigma(-15)$
k^{th} b,	$n-k$	$-2/3$	1	$\sigma(-15)$
$k < n$		0	~ 1	~ 0
n^{th} b	0	$-2/3$	0	$\sigma(-15)$
		0	0	1

Table 2: Unit values (columns) during each phase of a valid $a^n b^n$ sequence (rows). The second line for output units, given in bold, indicates the final normalized probability.

observed during the b 's). The network not only accepts valid $a^n b^n$ sequences, but also rejects other sequences, visible by the zero probability it assigns to irrelevant outputs at each phase in Table 2.

More informally, the network uses a single hidden unit (6) as a counter, recognizable from the recurrent loop onto itself. The counter is incremented by 1 for each a (+2 from unit 1, -1 from the bias), and then decremented by 1 for each b (signaled by a lack of a , which leaves only the -1 bias as input to the counter). \square

Theorem 4.2. *The network represented in Fig. 4 outputs the correct probability for each character, for each sequence in the $a^n b^n c^n$ language, with a margin of error below 10^{-6} .*

Proof. The proof is again obtained by tracing the values each unit holds at each phase of a valid sequence in the language, see Table 3.

The network uses two hidden units that serve as counters for the number of a 's (unit 8) and c 's (unit 9). Each occurrence of a simultaneously feeds the output unit for a (5) and the a counter (8) connected to the b output (6), using weights

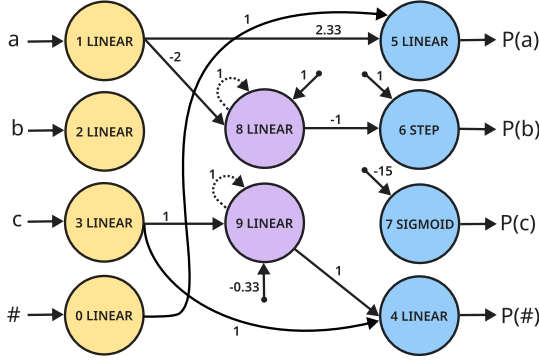


Figure 4: The network found for the $a^n b^n c^n$ task for the larger training set. See Theorem 4.2 for a description of how this network accepts only sequences of the language $a^n b^n c^n$.

$a^n b^n c^n$	Unit 8	Unit 9	Unit 4	Unit 5	Unit 6	Unit 7
			$P(\#)$	$P(a)$	$P(b)$	$P(c)$
Initial #	1	$-\frac{1}{3}$	$-\frac{1}{3}$	1	0	$\sigma(-15)$
k^{th} a	$1 - k$	$-\frac{k+1}{3}$	$-\frac{k+1}{3}$	$\frac{7}{3}$	1	$\sigma(-15)$
			~ 0	$\sim .7$	$\sim .3$	~ 0
k^{th} b	$k+1-n$	$-\frac{k+n+1}{3}$	$-\frac{k+n+1}{3}$	0	1	$\sigma(-15)$
$k < n$			0	0	~ 1	~ 0
n^{th} b	1	$-\frac{2n+1}{3}$	$-\frac{2n+1}{3}$	0	0	$\sigma(-15)$
			0	0	0	1
k^{th} c	$1+k$	$\frac{2k-2n-1}{3}$	$\frac{2(k+1-n)}{3}$	0	0	$\sigma(-15)$
$k < n$			0	0	0	1
n^{th} c	$1+n$	$-\frac{1}{3}$	$\frac{2}{3}$	0	0	$\sigma(-15)$
			~ 1	0	0	~ 0

Table 3: Unit values (columns) during each phase of a valid $a^n b^n c^n$ sequence (rows).

to create the correct probability distribution between a 's and b 's. Once a 's stop, $P(a)$ flatlines, and the a counter (8) starts decreasing until n b 's are seen. Another counting system has evolved in unit 9 which counts the number of a 's and b 's (signaled by lack of c 's), and then decreases for each c , finally triggering the end-of-sequence output $\#$. Note how the model avoids holding a third counter for the number of b 's, by reusing the a counter. This makes it possible to disconnect the b input unit (2), which minimizes encoding length. \square

Theorem 4.3. *The $a^n b^{2n}$ network represented in Fig. 5 outputs the correct probability for each character, for each sequence in the $a^n b^{2n}$ language, with a margin of error below 10^{-6} .*

Proof. The network is similar to the one found for $a^n b^n$ (Fig. 3). The proof that this network is accurate is also similar (Theorem 4.1), the only difference being that the hidden unit is incremented with 2 instead of 1 for each a input. \square

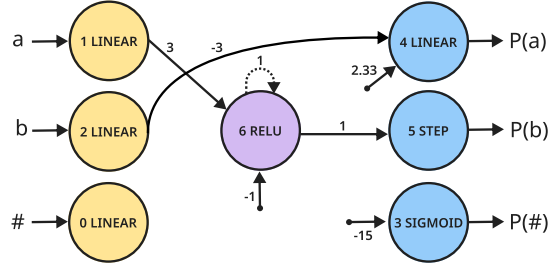


Figure 5: The network found for the $a^n b^{2n}$ task for the larger training set. See Theorem 4.3 for a description of how this network accepts only sequences of the language $a^n b^{2n}$.

Theorem 4.4. *The network represented in Fig. 6 outputs the correct probability for each character, for each sequence in the $a^n b^m c^{n+m}$ language, with a margin of error below .2 (and below 10^{-4} for deterministic steps, i.e., probabilities 0 or 1).⁶*

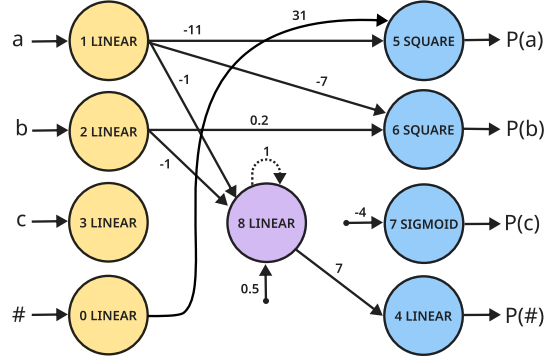


Figure 6: The network found for the $a^n b^m c^{n+m}$ task for the larger training set. See Theorem 4.4 for a description of how this network accepts only sequences of the language $a^n b^m c^{n+m}$.

Proof. In Table 4 we trace the values of each unit during feeding of a valid sequence in $a^n b^m c^{n+m}$. We do not represent the internal memory unit 8, its value is the seventh of that of unit 4.

Here, a network with a single counter (unit 8) has evolved which recognizes the language with 100% accuracy. While one would think that this task requires at least two counters — for n and m — the pressure for parsimony leads the model to evolve a more compact solution: since the number of c 's is always $n + m$, and no other symbols

⁶For this task, the average test cross-entropy per character of the network trained on the larger data set goes slightly below the optimum (see Table 1); this can happen for example if the model picks up on unintentional regularities in the training set that are also present in the test set.

$a^n b^m c^{n+m}$	unit 5 $P(a)$	unit 6 $P(b)$	unit 7 $P(c)$	unit 4 $P(\#)$
Initial #	31^2 $\sim .996$	0	$\sigma(-4)$ ~ 0	$\frac{7}{2}$ $\sim .004$
k^{th} a	11^2 $\sim .71$	7^2 $\sim .29$	$\sigma(-4)$ ~ 0	$\frac{7}{2}(1-k)$ 0
k^{th} b	0 0	.04 $\sim .69$	$\sigma(-4)$ $\sim .31$	$\frac{7}{2}(1-n-k)$ 0
k^{th} c	0	0	$\sigma(-4)$	$\frac{7}{2}(1-n-m+k)$
$k < m + n$	0	0	1	0
$(m + n)^{\text{th}}$ c	0	0	$\sigma(-4)$ ~ 0	$\frac{7}{2}$ ~ 1

Table 4: Unit values during each phase of a valid $a^n b^m c^{n+m}$ sequence.

appear between the first and last #, the network uses the signal of *lack of c's* as an indication of *positive occurrences* of either *a* or *b*. This might raise a suspicion that the network recognizes out-of-language sequences such as balanced yet unordered strings, e.g. *abbaaccccc*. In practice, however, the network imposes a strict order: *a* receives a positive probability only after # or *a*; *b* only after *a* or *b*; and *c* receives a significant proportion of the probability mass only as a last resort. \square

4.4 Experiment II: Dyck-1 vs. Dyck-2

In previous tasks, we showed the capability of MDLRNNs to evolve counters. A counter is also what is needed to recognize the Dyck-1 language of well-matched parentheses sequences. In the Dyck-2 language, however, there are two pairs of opening and closing characters, such as parentheses and brackets. Counters are not sufficient then, and a stack is needed to additionally track whether the next closing element must be a parenthesis or a bracket (and similarly for any Dyck- n language for $n > 1$, [Suzgun et al., 2019](#)). We ask here whether MDL-optimized networks can evolve not only counters but also stacks.

4.4.1 Setup

The setup is that of a language modeling task, as in Experiment I. For Dyck-1, the training sequences were generated from a PCFG with probability $p = .3$ of opening a new bracket, with data set sizes 100 and 500. The test sets contained 50,000 sequences generated from the same grammar that were not seen during training.

For Dyck-2, a fully operational stack is needed in order to recognize the language. We thus first make sure that such a network exists in the search space. We do this by manually designing a network that implements a fully operational stack.

We use this network as a baseline for comparison with the results of the MDL simulations.

The stack network and a detailed description of its mechanism are given in Fig. 8. We add two additional building blocks in order to implement this mechanism: the modulo 3 activation function used in the ‘pop’ implementation, and a second type of unit which applies multiplication to its inputs, in order to create gates such as the ones used in LSTM networks. Because the inclusion of these new representations enlarges the search space, and because the baseline network is larger in terms of number of units than the networks found in previous tasks (23 vs. 7-10), we double the genetic algorithm’s overall population size (500 islands vs. 250), allowing more hypotheses to be explored. We also enlarge the training set to 20,000 samples, which allows networks with costlier $|G|$ terms to evolve. Here again the training sequences were generated from a PCFG with probability $p = .3$ for opening a new bracket or parenthesis, and tested on 50,000 novel sequences generated from the same grammar.

Dyck sequences don’t have any sub-parts which can be predicted deterministically (one can always open a new bracket), which makes *deterministic accuracy* reported above irrelevant. We report instead a metric we call **categorical accuracy**, defined as the fraction of steps where the network predicts probability $p \geq \epsilon$ for symbols that *could* appear at the next step, and $p < \epsilon$ for irrelevant symbols. For example, for Dyck-2, when the upcoming closing character is a bracket (i.e., the last seen opening character was a bracket), the network should assign probability 0 to the closing parenthesis; and similarly for the end-of-sequence symbol as long as a sequence is unbalanced. Because classical RNNs cannot assign categorical 0 probabilities to outputs due to their reliance on softmax layers, we use $\epsilon = 0.005$ as a categorical margin.

4.4.2 Results

Full performance details are given in Table 1.

For the Dyck-1 language, the networks for the small and large training sets reach average test cross-entropy of 1.11 and 0.89 respectively, compared to an optimal 0.88. This result is in line with those of Experiment I, where we have shown that our representations are capable of evolving counters, which are sufficient for recognizing Dyck-1 as well. An Elman RNN reaches a better cross-entropy score, but worse categorical accuracy, for

the smaller training set, while MDLRNN wins with the larger set, reaching a score close to the optimum and 100% categorical accuracy.

Theorem 4.5. *When brackets are well balanced, the Dyck-1 network in Fig. 7 correctly predicts that no closing bracket can follow by assigning it probability 0. Conversely, when brackets are unbalanced, it assigns probability 0 to the end-of-sequence symbol.*

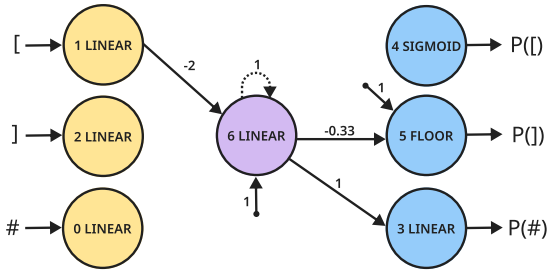


Figure 7: The network found by the MDL learner for the Dyck-1 task for the larger training set. See Theorem 4.5 for a description of how it accepts only valid Dyck-1 sequences.

Dyck-1	Unit 6	Unit 4 $P([)$	Unit 5 $P([)$	Unit 3 $P(\#)$
$o > 0$	$1 - o$	$1/2$	$\text{floor}(\frac{2+o}{3})$	$1 - o$
		$\sim \frac{3}{7+2o}$	$\sim \frac{4+2o}{7+2o}$	0
$o = 0$	1	$1/2$	$\text{floor}(\frac{2}{3}) = 0$	1
		1/3	0	2/3

Table 5: Unit values and output probabilities during the two possible phases of a Dyck-1 sequence: (i) the number of open brackets o is positive, or (ii) all brackets are well balanced ($o = 0$).

Proof. Call o the number of open brackets in a prefix. Throughout a Dyck-1 sequence, unit 6 holds the value $1 - o$: it holds the value 1 after the initial '#'; then +1 is added for each '[', and -1 for each ']'. The output probabilities in the cases of balanced and unbalanced sequences are then given in Table 5. The theorem follows from the fact that $P(\#) = 0$ if $o > 0$, and $P([) = 0$ if $o = 0$. In the target language, we note that opening brackets have a constant probability of $P([) = .3$, while in the found network this probability decreases with o (visible in unit 4’s output probability, Table 5). This makes a potential difference for high values of o , which however are very rare (o decreases with probability .7 at all time steps). \square

For Dyck-2, the MDL model fails to reach the architecture of the baseline manual network, or another architecture with a similar cross-entropy score, reaching a network which has a worse MDL score than the baseline (148,497 vs. 147,804). Accordingly, MDLRNN reaches a non-perfect 99.27% categorical accuracy, compared to 89.01% for RNNs, which reflects both models’ failure to correctly balance certain sequences. Both models tie at 1.19 cross-entropy, close to the optimal 1.18.

Since we confirmed that the baseline architecture exists in the search space, we conclude that reaching a fully operational stack network is hindered by the non-exhaustive search procedure, rather than by the MDL metric. This may be solvable by tweaking the hyper-parameters or putting more computational resources into the search. It could be, however, that this difficulty is due to a more interesting property of the task at hand. It has been claimed that evolutionary algorithms tend to struggle with so-called ‘deceptive’ optimization problems — tasks for which series of intermediate good solutions don’t necessarily lead to a global optimum (see overview in Lehman and Stanley, 2011). For the stack network, it could be the case that a stack is only operational in its full form, and that intermediate networks deceive and lead the search to local minima, like the one found in the current simulation.

A recent line of work has addressed the need for stacks by manually designing stack-like mechanisms using continuous representations, and integrating them manually into standard architectures (Graves et al., 2014, Joulin and Mikolov, 2015, Suzgun et al., 2019, among others). Indeed, when they are explicitly augmented with manually-designed continuous stack emulators, neural networks seem to be able to capture non-regular grammars such as the one underlying the Dyck-2 language. Similarly, we could allow our search to add stacks in one evolution step. This could overcome the risk of a deceptive search target mentioned above. If successful, we can expect this approach to come with all the benefits of the MDL approach: the winning network would remain small and transparent, and it would eventually contain a memory stack only if this is intrinsically needed for the task.

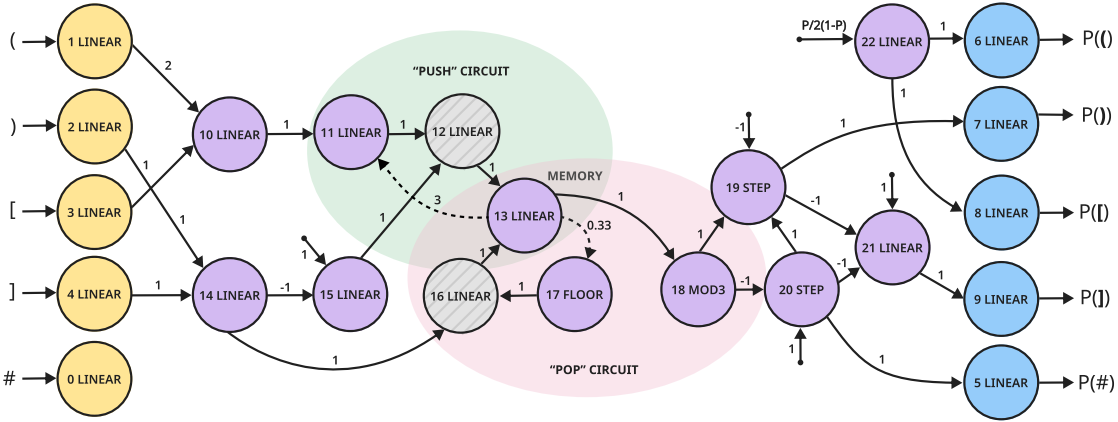


Figure 8: A manually-designed network implementing a fully operational stack, which recognizes the Dyck-2 language. The network uses an additional type of unit, which calculates the product of its inputs instead of summing them, making it possible to create gate units similar to those of LSTM networks (gray striped units in the figure). The stack’s memory is implemented as an integer, stored here in unit 13; the integer is shifted to the left or right in base 3, making it possible to store the value 2 for a parenthesis and 1 for a bracket, visible in their respective input weights. Unit 12 is the ‘push’ gate, which opens when a non-zero value flows from the opening bracket or parenthesis inputs. Unit 16 is the ‘pop’ gate, opened by a non-zero input from a closing symbol. The recurrent connection from memory unit 13 to unit 11 performs the base-3 left shift by multiplying the memory by 3. For ‘pop’, a right shift is applied by dividing the memory by 3. To extract the value of the topmost element, modulo 3 is applied. The bias for unit 22 handles outputting the probability p of opening a new bracket/parenthesis.

4.5 Experiment III: general addition

In the previous experiments, we saw that MDL-optimized networks are capable of representing integers and add them in what amounts to unary representation (see $a^n b^m c^{n+m}$ language). Here, we show that addition can be performed when the numbers and outputs are represented in a different format. Specifically, we consider the familiar task of adding two integers in binary representation when the numbers are fed bit-by-bit in parallel, starting from the least significant bit. While this problem has good approximate solutions in terms of standard RNNs,⁷ we will show that our model provides an *exact* solution. As far as we are aware, this has not been shown before.

4.5.1 Setup

In this setting, we diverge from a language modeling task. The network here is fed at each time step i with a tuple of binary digits, representing the digits n_i and m_i of two binary numbers n and m , starting from the least significant bit. The two input units are assigned the values n_i and m_i . The

⁷An example implementation that reportedly works up to a certain number of bits: https://github.com/mineshmathew/pyTorch_RNN_Examples

output is interpreted as the predicted probability that $(n + m)_i = 1$, that is that 1 is the i^{th} digit in the sum $(n + m)$. Output values are capped to make them probabilities: values at or below 0 are interpreted as probability 0, values at or above 1 are interpreted as probability 1.

The model was trained on two corpus sizes: one that contained all pairs of integers up to $K = 10$ (total 100 samples), and a larger set of all pairs up to $K = 20$ (total 400). The resulting networks were then tested on the set of all pairs of integers $n, m \in [K + 1, K + 251]$, i.e., 62,500 pairs not seen during training. Since the task is fully deterministic, we report a standard accuracy score.

4.5.2 Results

MDLRNNs reached 100% accuracy on both test sets, and an optimal cross-entropy score of zero. Fig. 9 shows the MDLRNN result for the larger training set. It probably does perfect addition, with perfect confidence, for all pairs of integers:

Theorem 4.6. *For the net in Fig. 9, the output unit at time step i is the i^{th} digit of the sum of the inputs.*

Proof. Call c_{i-1}^3 the value of unit 3 at time step $i - 1$; this value is the carry-over for the next time

step, feeding unit 4 through their recurrent connection at time step i . This can be proven in two steps. (1) At the first time step $i = 1$ the carry-over going into unit 4 is 0, since recurrent inputs are 0 by default at the first time step. (2) By induction, c_i^4 is the sum of the relevant carry-over (c_{i-1}^3) and the two input digits at time i . The combination of the $1/2$ multiplication and floor operation extracts a correct carry-over value from that sum and stores it in unit 3. From there, we see that c_i^2 holds the correct binary digit: the sum of current inputs and carry-over (from c_i^4), minus the part to be carried over next (from $-2 \times c_i^3$). \square

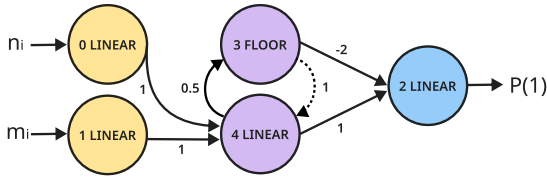


Figure 9: The network found by the MDL learner for the binary addition task, trained on all 400 pairs of numbers up to 20. This network is correct for all numbers (Theorem 4.6).

Again, the task is learned perfectly and in a readable fashion. As a side remark, the network obtained here can also naturally be extended to perform addition of more than 2 numbers, simply by adding the necessary inputs for the additional digits and connecting them to cell 4. To our knowledge no other RNN has been proven to hold a carry-over in memory for an unbounded number of digits, i.e. to perform general addition of any arbitrary pair of numbers. The best competing classical RNNs trained here were never able to reach more than 79.4% accuracy on the test sets, indicating that they learned a non-general way to do addition.

4.6 Objective function probe

In order to further probe the value of the MDL objective function — and to isolate the effects of the objective function, which is our main focus, from those of the training method and the activation functions — we ran four additional simulations using variations of MDL while keeping the setting without change. The variants of the objective function that we tested are: (i) $|G|$ alone, i.e., only the description length of the network is minimized; (ii) $|D : G|$ alone, i.e., the model only opti-

mizes training data fit, similarly to a cross-entropy loss in traditional models; (iii)-(iv) replacing $|G|$ with traditional L1 and L2 regularization terms.

The different objective functions were tested on the $a^n b^n$ task using the same hyper-parameters given in Sec. 4.1. Table 6 summarizes the performance for each resulting network. As expected, when $|G|$ alone is minimized, the result is a degenerate network with no hidden units or connections. Conversely, $|D : G|$ -only training results in a network growing large and picking up on accidental regularities in the training set. The overfitting leads to below-optimal cross-entropy on the training set. Test cross-entropy is infinite because the model assigns a categorical zero probability to some possible targets. Both L1 and L2 regularizations indirectly constrain the encoding length of the resulting networks and have the advantage of being compatible with backpropagation search. However, these constraints are not as effective as pure MDL in avoiding overfitting (cross-entropy is below optimal on the training set and above on the test set).

Objective function	CE ($\times 10^{-2}$)		Size	
	train	test	units	conn
$ G $	158.5	158.5	0	0
$ D : G $	37.3	∞	126	299
$ D : G + L1$	37.6	55.3	6	23
$ D : G + L2$	37.5	∞	6	33
$ D : G + G $ (MDL)	38.1	25.8	1	7

Table 6: Cross-entropy and number of units and connections on the $a^n b^n$ task using different objective functions; MDL yields ground-truth optimal CE for both training and test.

5 Conclusion

Classical RNNs optimized for accuracy can partially recognize nonregular languages and generalize beyond the data up to a certain n (Gers and Schmidhuber, 2001; Weiss et al., 2018b). However large this n may be, the failure of these networks to fully generalize to arbitrary values of n reveals that they fail to lock in on the correct grammars that underlie these tasks.

We found that an MDL-optimized learner arrives at networks that are reliably close to the true distribution with small training corpora, for classically challenging tasks. In several cases, the networks achieved perfect scores. Beyond the usual evaluation in terms of performance on test sets, the

networks lent themselves to direct inspection and showed an explicit statement of the pattern that generated the corpus.

Acknowledgements

We wish to thank Matan Abudy, Moyshe Bar-Lev, Artyom Barmazel, Marco Baroni, Adi Behar-Medrano, Maxime Cauté, Rahma Chaabouni, Emmanuel Dupoux, Nicolas Guérin, Jean-Rémy King, Yair Lakretz, Tal Linzen, Aël Quelellennec, Ezer Rasin, Mathias Sablé-Meyer, Benjamin Spector; the audiences at CNRS/ENS Paris, Facebook AI Paris, NeuroSpin, Tel Aviv University, and ZAS Berlin; and Nitzan Ron for creating the figures in this paper. We also thank the action editors at TACL and three anonymous reviewers for their helpful comments.

This work was granted access to the HPC/AI resources of IDRIS under the allocation 2021-A0100312378 made by GENCI.

References

- Panagiotis Adamidis. 1994. Review of parallel genetic algorithms bibliography. *Aristotle Univ. Thessaloniki, Thessaloniki, Greece, Tech. Rep.*
- Fardin Ahmadizar, Khabat Soltanian, Fardin AkhlaghianTab, and Ioannis Tsoulos. 2015. Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm. *Engineering Applications of Artificial Intelligence*, 39:1–13.
- P.J. Angeline, G.M. Saunders, and J.B. Pollack. 1994. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65.
- Robert C. Berwick. 1982. *Locality Principles and the Acquisition of Syntactic Knowledge*. Ph.D. thesis, MIT, Cambridge, MA.
- Erick Cantú-Paz. 1998. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171.
- Gregory J. Chaitin. 1966. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. [Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation](#). *arXiv:1406.1078 [cs, stat]*.
- Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.
- Adam Gaier and David Ha. 2019. Weight agnostic neural networks. *CoRR*, abs/1906.04358.
- Felix Gers and Jürgen Schmidhuber. 2001. [LSTM recurrent networks learn simple context-free and context-sensitive languages](#). *IEEE Transactions on Neural Networks*, 12(6):1333–1340. Conference Name: IEEE Transactions on Neural Networks.
- C. Lee Giles, Guo-Zheng Sun, Hsing-Hen Chen, Yee-Chun Lee, and Dong Chen. 1990. Higher order recurrent networks and grammatical inference. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 380–387. Morgan-Kaufmann.
- V Scott Gordon and Darrell Whitley. 1993. Serial and parallel genetic algorithms as function optimizers. In *ICGA*, pages 177–183.
- Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. [Neural Turing Machines](#). *arXiv:1410.5401 [cs]*. ArXiv: 1410.5401.
- Peter Grünwald. 1996. A minimum description length approach to grammar inference. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, Springer Lecture Notes in Artificial Intelligence, pages 203–216. Springer.
- Geoffrey E. Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780. Publisher: MIT Press.
- John H Holland. 1975. Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*, pages 439–444.

- James Horning. 1969. *A Study of Grammatical Inference*. Ph.D. thesis, Stanford.
- Henrik Jacobsson. 2005. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, 17(6):1223–1263.
- Armand Joulin and Tomas Mikolov. 2015. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A Method for Stochastic Optimization](#). In *International Conference of Learning Representations (ICLR)*.
- Andrei Nikolaevic Kolmogorov. 1965. Three approaches to the quantitative definition of information. *Problems of Information Transmission (Problemy Peredachi Informatsii)*, 1:1–7.
- Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. [The emergence of number and syntax units in LSTM language models](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.
- Joel Lehman and Kenneth O. Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223. Publisher: MIT Press.
- Ming Li and Paul Vitányi. 2008. [Chapter 1.4, Binary Strings](#). In *An Introduction to Kolmogorov Complexity and Its Applications*, Texts in Computer Science. Springer New York, New York, NY.
- Carl de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, MIT, Cambridge, MA.
- Geoffrey F Miller, Peter M Todd, and Shailesh U Hegde. 1989. *Designing Neural Networks using Genetic Algorithms.*, volume 89.
- David J Montana and Lawrence Davis. 1989. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767.
- Ezer Rasin, Iddo Berger, Nur Lan, Itamar Shefi, and Roni Katzir. 2021. Approaching explanatory adequacy in phonology using Minimum Description Length. *Journal of Language Modelling*, 9(1):17–66.
- Ezer Rasin and Roni Katzir. 2016. On evaluation metrics in Optimality Theory. *Linguistic Inquiry*, 47(2):235–282.
- Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.
- Jürgen Schmidhuber. 1997. Discovering neural nets with low Kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873.
- Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61(0):85–117.
- Ray J. Solomonoff. 1964. A formal theory of inductive inference, parts I and II. *Information and Control*, 7(1 & 2):1–22, 224–254.
- Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127. Publisher: MIT Press.
- Andreas Stolcke. 1994. *Bayesian Learning of Probabilistic Language Models*. Ph.D. thesis, University of California at Berkeley, Berkeley, California.
- Mirac Suzgun, Sebastian Gehrmann, Yonatan Belinkov, and Stuart M. Shieber. 2019. [Memory-Augmented Recurrent Neural Networks Can Learn Generalized Dyck Languages](#). *arXiv:1911.03329 [cs]*.
- Qinglong Wang, Kaixuan Zhang, Alexander G. Ororbis II, Xinyu Xing, Xue Liu, and C. Lee Giles. 2018. An empirical evaluation of rule extraction from recurrent neural networks. *Neural Computation*, 30(9):2568–2591.

- Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018a. Extracting automata from recurrent neural networks using queries and counterexamples. In *Proceedings of the 35th International Conference on Machine Learning*.
- Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018b. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745.
- D Whitley, T Starkweather, and C Bogart. 1990. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14(3):347–361.
- Yuan Yang and Steven T. Piantadosi. 2022. [One model for the learning of language](#). *Proceedings of the National Academy of Sciences*, 119(5). Publisher: National Academy of Sciences Section: Social Sciences.
- Byoung-Tak Zhang and Heinz Mühlenbein. 1993. Evolving optimal neural networks using genetic algorithms with Occam’s Razor. *Complex Systems*, 7(3):199–220.
- Byoung-Tak Zhang and Heinz Mühlenbein. 1995. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation*, 3(1):17–38.

Chapter 2

Benchmarking Neural Network

Generalization for Grammar Induction

(joint with Emmanuel Chemla and Roni Katzir)

Nur Lan, Emmanuel Chemla, and Roni Katzir. 2023. Benchmarking Neural Network Generalization for Grammar Induction. In *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*, pages 131–140, Gothenburg, Sweden. Association for Computational Linguistics. <https://aclanthology.org/2023.clasp-1.15>

Benchmarking Neural Network Generalization for Grammar Induction

Nur Lan^{1,2}, Emmanuel Chemla¹, Roni Katzir²

¹Ecole Normale Supérieure

²Tel Aviv University

{nur.lan, emmanuel.chemla}@ens.psl.eu

rkatzir@tauex.tau.ac.il

Abstract

How well do neural networks generalize? Even for grammar induction tasks, where the target generalization is fully known, previous works have left the question open, testing very limited ranges beyond the training set and using different success criteria. We provide a measure of neural network generalization based on fully specified formal languages. Given a model and a formal grammar, the method assigns a generalization score representing how well a model generalizes to unseen samples in inverse relation to the amount of data it was trained on. The benchmark includes languages such as $a^n b^n$, $a^n b^n c^n$, $a^n b^m c^{n+m}$, and Dyck-1 and 2. We evaluate selected architectures using the benchmark and find that networks trained with a Minimum Description Length objective (MDL) generalize better and using less data than networks trained using standard loss functions. The benchmark is available at <https://github.com/taucompling/bliss>.

1 Introduction

The extent to which artificial neural networks (ANNs) generalize beyond their training data is an open research question. In this work we approach this question from the perspective of grammar induction, that is, the learning of a formal grammar from a finite (often small) sample from the (typically infinite) language of that grammar. In order to succeed in this task, a model must strike a balance between fitting the training data and generalizing to a potentially infinite set of unseen strings. Humans tested on such tasks show systematic generalization from small sets of examples (Fitch and Hauser, 2004, Malassis et al., 2020).

While a range of ANN architectures have been shown to reach approximations for formal languages, the quality of this approximation remains an open matter, as we show below. Here we build on previous probes of ANN generalization for grammar induction and introduce a unified and

general way to assess this capability, for a given pair of a learning model and a corpus drawn from a formal language. Our main contributions are:

1. **A benchmark for formal language learning.** The benchmark relies on a method for quantifying ANN generalization for formal languages, including probabilistic languages. The method assigns an index score representing a model’s generalization performance in inverse relation to the size of the training data. We introduce the method and provide concrete datasets for well-studied formal languages.
2. **An evaluation of selected architectures.** We test recurrent neural networks (RNNs) of the Long-Short Term Memory type (LSTM; Hochreiter and Schmidhuber, 1997); Memory-augmented RNNs (MARNN; Suzgun et al., 2019b;) and an RNN variant which replaces the traditional gradient-based training regime with an objective that optimizes the model’s Minimum Description Length (MDLRNN; Lan et al., 2022).

We find that equipping ANNs with memory devices such as differentiable stacks helps generalization, but generalization remains partial and slow. At the same time, training with MDL leads in some of the test cases that we examined to potentially perfect generalization with significantly less data. In other cases, training with MDL did not result in successful generalization, possibly because the optimization procedure we used for the architecture search failed to find the global optimum under the MDL objective function.

2 Background

Learning formal languages has long been used to probe various aspects of ANN performance. These most often include inquiries about: (i) ANNs’ ability to generalize beyond their training data, and

Language	Paper	Model	Metric	Training size	Max train n	Max test n
$a^n b^n$	GS'01	LSTM	$M_{cat'}$	16,000	30	1,000
	JM'15	Stack-RNN	M_{det}	20 [†]	19	60
	WGY'18	LSTM	Bin	100 [†]	100	256
	LGCK'22	MDLRNN	M_{det}	500	22	∞
$a^n b^n c^n$	GS'01	LSTM	$M_{cat'}$	51,000	40	500
	JM'15	Stack-RNN	M_{det}	20 [†]	19	60
	WGY'18	LSTM	Bin	50 [†]	50	100
	LGCK'22	MDLRNN	M_{det}	500	22	∞
Dyck-1	SGBS'19a	LSTM	$M_{cat'}$	10,000	50	100
	SGBS'19b	MARNN	$M_{cat'}$	5,000	50	100
	EMW'22	ReLU-RNN	$M_{cat'}$	10,000	50	1,000
	LGCK'22	MDLRNN	M_{cat}	500	16	∞

Table 1: ANN performance in selected probes of formal language learning. **Metrics** (see Section 3.5): M_{det} = deterministic accuracy; M_{cat} = categorical accuracy; $M_{cat'}$ = a non-probabilistic version of M_{cat} ; Bin = binary classification from hidden state to accept/reject labels, based on positive and negative samples. **Training size**: † = the paper did not explicitly specify the training set size, the value here is derived by assuming the training set was an exhaustive list of all strings up to ‘max train n ’. ‘**Max test n** ’: the largest n for which the criterion was reached. For Dyck-1, n represents overall sequence length. ‘ ∞ ’ = the paper provides evidence that the network is correct for any n . When a paper reports several experiments as in GS'01, we take the best result based on the smallest training set. **Papers**: GS'01 = Gers and Schmidhuber (2001); JM'15 = Joulin and Mikolov (2015); WGY'18 = Weiss et al. (2018); SGBS'19a = Suzgun et al. (2019a); SGBS'19b = Suzgun et al. (2019b); EMW'22 = El-Naggar et al. (2022); LGCK'22 = Lan et al. (2022).

(ii) ANNs' expressive power; i.e., whether they can represent the relevant target grammars (often probed with reference to the Chomsky hierarchy of formal languages, as in Delétang et al., 2022). Here we will focus on the generalization question. We will show how it might be related to another under-exploited line of inquiry regarding the training objective of ANNs.

A long line of theoretical work has probed the computational power of ANNs. Siegelmann and Sontag (1992) originally showed that RNNs with a sigmoid activation can emulate multiple-stack Turing machines under certain permissive conditions (infinite activation precision and unbounded running time). Since these conditions cannot be met in practice, another line of work probed the computational power of RNNs under practical conditions (finite precision and input-bound running time). Weiss et al. (2018) have shown that under these conditions LSTMs are able to hold weight configurations that perform unbounded counting, and so they should be able to recognize counter languages (CL), a family of formal languages that can be recognized using one or more counting devices (following some formal restrictions, Merrill, 2021). Recently, El-Naggar et al. (2023a) and El-Naggar et al. (2023b) have shown that two simpler RNN ar-

chitectures, with linear- and ReLU-based cells, are also able to hold counting weight configurations, with similar consequences for recognizing CL.

Empirically, another line of work provided promising results regarding the capability of ANNs to learn formal languages. This was most often done by training networks on strings up to a certain length and then showing good performance on longer ones (Bodén and Wiles, 2000, Gers and Schmidhuber, 2001; see Table 1). Gers and Schmidhuber (2001) have shown that LSTMs trained on languages such as $a^n b^n$ and $a^n b^n c^n$ with n values in the low dozens perform well on n 's in the high hundreds. Suzgun et al. (2019a) found that LSTMs trained on Dyck-1 sequences (strings of well-balanced pairs of brackets) up to length 50 performed well on lengths up to 100. Suzgun et al. (2019b) proposed RNN variants that are equipped with external differentiable memory devices and showed that they yield improved performance on non-regular languages.

However, other empirical results show that in practice ANNs generalize only to very restricted ranges. Weiss et al. (2018) found that while LSTMs are theoretically able to hold counting solutions, these are not found through training: LSTMs trained on $a^n b^n$ and $a^n b^n c^n$ with max n 100 and

50, respectively, start accepting illicit strings with n values as low as 256 and 100. As mentioned above, [Suzgun et al. \(2019a\)](#) tested LSTMs on Dyck-1 sequences but only up to length 100, and concluded that this language was learned by LSTMs. [El-Naggar et al. \(2022\)](#) extended this work to longer sequences, and found that LSTMs fail to generalize in practice, outputting incorrect predictions at lengths under 1,000. This, despite Dyck-1 being a CL and so theoretically learnable by LSTMs ([Weiss et al., 2018](#)).

Apart from LSTMs, recent probes by [El-Naggar et al. \(2023a\)](#) and [El-Naggar et al. \(2023b\)](#) have shown that linear and ReLU RNNs, theoretically capable of counting, fail to find the counting weight configurations in practice when trained using back-propagation and standard loss functions; [El-Naggar et al. \(2023b\)](#) went further with determining the source of this discrepancy, showing that the counting weight configuration is not an optimum of these loss functions.

Moreover, even in works that report successful generalization to some degree beyond the training set, the fact that networks stop generalizing at an arbitrary point is often left unexplained ([Gers and Schmidhuber, 2001](#), [Suzgun et al., 2019a, 2019b](#), [Delétang et al., 2022](#), a.o.).¹

The literature on the generalization abilities of ANNs has made use of a range of measures of success, making results difficult to compare. Different probes of the same model often use different success criteria, and generate training and test sets using different sampling methods and of different orders of magnitude. [Table 1](#) summarizes selected probes of ANN generalization and highlights the fragmented nature of this literature. In the following sections we propose a unified method to consolidate these efforts and better understand the generalization capabilities of ANNs.

3 The BLISS index

We present the Benchmark for Language Induction from Small Sets (BLISS). We provide a formal description of the method, followed by a concrete application to specific tasks.

¹Technical limitations such as finite activation precision can be ruled out as explanations for generalization failures, at least for counter languages and models where network states serve as memory: as shown in works mentioned above, ANNs often start outputting wrong predictions for n values in the low hundreds. Even restricted representations such as 16-bit floats can hold much larger values, and modern implementations such as PyTorch use 32-bit floats by default.

The current release consists of three parts: (i) A specification for the generalization index \mathbb{B} , calculated for a given pair of formal language and ANN; (ii) A dataset containing a set of formal languages for benchmarking; (iii) An evaluation of different ANN architectures using this dataset.

3.1 General setting: models and tasks

For a given model A , e.g., an LSTM, a task is composed of the following components:

- G – a grammar, e.g., a probabilistic context-free grammar (PCFG).
- S – a sampling method from $\mathcal{L}(G)$, the language generated by G .
- $\mathcal{C} = S(G)$ – a training corpus, may contain repetitions.
- $\mathcal{T} \subseteq \mathcal{L}(G) \setminus \mathcal{C}$ – a test corpus.
- M – a task-specific accuracy metric with adjustable error margin

$\varepsilon \in [0, 1]$. It uses predictions $A(s)$ on strings $s \in \mathcal{T}$ to calculate an accuracy score $M(A, \mathcal{T}, \varepsilon) \in [0, 1]$.

- N – a task-specific constant for setting the order of magnitude of dataset sizes. For example, $N = 3$ sets the order of magnitude at 10^3 . Training and test sizes are then derived as described below. Selecting N is done empirically based on properties of the task, e.g., languages with large vocabularies require larger amounts of training data, hence a larger N .

3.2 From task to generalization index

For a given task, the generalization index of order N for a model A is then defined as:

$$\mathbb{B}_N^{\mathcal{L}}(A) = \max \left\{ b \mid \begin{array}{l} |\mathcal{T}| = 10^N \times b, \\ |\mathcal{C}| = 10^N / b, \\ M(A, \mathcal{T}, \varepsilon) = 1.0 \end{array} \right\} \quad (1)$$

Intuitively, the index compares a model’s performance on a test size $|\mathcal{T}|$ to the inverse of its training data size $|\mathcal{C}|$.

The index is expressed as the maximal b factor which scales the training set and the corresponding test set in opposite directions: The accuracy condition at the bottom of (1) means that the model should be ε -close to perfect generalization on the test set. A model’s generalization index \mathbb{B} thus represents the performance that can be maximally ‘squeezed out’ of an inversely small amount of data.

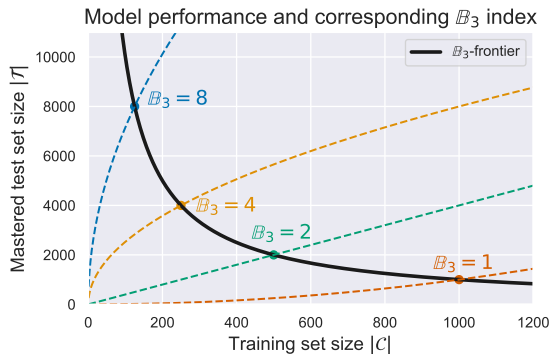


Figure 1: Example generalization index scores \mathbb{B}_3 , i.e., for a baseline training size of 10^3 . Each dashed line represents the performance profile of some hypothetical model, as a function of the size of the training set. The intersection with the \mathbb{B}_3 -frontier indicates its \mathbb{B}_3 index.

Figure 1 exemplifies selected \mathbb{B} values calculated based on (1). For illustration, for $a^n b^n$, using the order of magnitude $N = 3$, a model that was trained on $|\mathcal{C}| = 10^3/2 = 500$ samples and was 100% accurate on a test set of size $10^3 \times 2 = 2,000$ will have an index score $\mathbb{B}_3^{a^n b^n} \geq 2$. A model for the same language that was trained on 250 samples only and generalized to a subsequent set of 4,000 samples will reach $\mathbb{B}_3^{a^n b^n} \geq 4$.

For practical reasons, one cannot exhaust all values of b to find \mathbb{B} . However, training and evaluating a model using a few b values is enough to reveal its generalization dynamics, as shown in experiments in Section 5. The following sections describe the specific choices made for the different benchmark components in these experiments.

3.3 Learning setup

Previous work surveyed here differed in their learning setup. Gers and Schmidhuber (2001) and Suzgun et al. (2019a, 2019b) trained networks in a non-probabilistic, supervised setup by exposing the model to all possible next symbols and minimizing the mean-squared error (MSE) – i.e., the model is given explicit information about the distribution of possible symbols. Joulin and Mikolov (2015) and Lan et al. (2022) used a setup that we adopt below, in which model outputs are probabilistic, and training is self-supervised language modeling (i.e., the model is exposed to the next symbol only) with a cross-entropy loss. Weiss et al. (2018) trained a binary classifier with accept/reject labels based on positive and negative examples.

Since our focus is grammar induction, here we

adopt the more demanding setup of learning from positive examples alone. All tasks are thus designed as self-supervised language modeling. At each time step, a model assigns a probability distribution to the next symbols in the string.

The benchmark is agnostic as to the internals of the model and its training, as long as its outputs represent a probability distribution over symbols. In practice, then, the method can be applied to any language model, not necessarily an ANN.

3.4 Sampling

To construct the training and test sets \mathcal{C} and \mathcal{T} we use the following as method S :

- To construct \mathcal{C} , we sample strings according to the distribution defined by G , with repetitions. For example, if G is a PCFG, it can be sampled by applying derivation rules chosen proportionally to their expansion probabilities. Repetitions are allowed so that \mathcal{C} follows a similar surface distribution to $\mathcal{L}(G)$ and so that the model can pick up on the underlying probabilities in G .
- To construct \mathcal{T} , we take the $|\mathcal{T}|$ subsequent strings starting right after the longest string in \mathcal{C} , sorted by length.² For example, for the language $a^n b^n$, if the longest string in the training set \mathcal{C} was $a^{17} b^{17}$, and the model needs to be tested on a set of 2000 strings, \mathcal{T} will be composed of the strings $a^{18} b^{18}, \dots, a^{2017} b^{2017}$.

The sampling method S can be either probabilistic as described here, or exhaustive, training on all strings in \mathcal{L} up to a certain length. We opt for probabilistic sampling because of the nature of the task at hand: the models under discussion here are trained to assign probabilities to the next symbol in a string, most often minimizing a cross-entropy loss. In practice, then, they always learn distributions over strings. Thus if \mathcal{C} follows a similar surface distribution to \mathcal{L} (given a large enough sample size), the model should eventually learn this distribution in order to minimize its loss.

Probabilistic sampling thus makes it possible to probe both a model’s knowledge about the surface forms of \mathcal{L} (by treating model outputs as categorical classes), and about their distribution. The modularity of the index makes it possible to choose

²Test strings may need to be sorted further according to specific properties of a language, see Section 4.1.

either option by varying the accuracy metric M , as we show in the next section.

3.5 Accuracy metrics

Ultimately we are interested in knowing whether a model accepts all strings in \mathcal{L} and rejects all others. In classical formal language theory, where discrete automata are used, acceptance is clear cut and taken as going into an accepting state. ANNs on the other hand use continuous representations with no standard acceptance criterion.

Different acceptance criteria have been used in previous works to measure success for ANNs: Gers and Schmidhuber (2001) and Suzgun et al. (2019b) defined acceptance of a string as a model assigning output values above a certain threshold to valid symbols only; Joulin and Mikolov (2015) measure accuracy at parts of strings that are completely predictable; and Weiss et al. (2018) turn a network into a recognizer by training a binary classifier from network states to accept/reject labels. Below we provide general versions of these accuracy metrics (omitting Weiss et al., 2018 who rely on negative examples).

Choosing which metric to use is based on the properties of the language at hand. Well-performing models might still deviate slightly from perfect accuracy due to practical limitations, such as a softmax function preventing a model from expressing categorical decisions. Thus for each

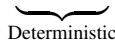
Input:	#	(()	())
	↓	↓	↓	↓	↓	↓	↓
Target:	#/((/	(/	(/	(/	(/	#/(
Input:	#	a	a	a	b	b	b
	↓	↓	↓	↓	↓	↓	↓
Target:	#/a	a/b	a/b	a/b	b	b	#
							

Figure 2: Inputs and valid next symbols at each step of a Dyck-1 string (top) and $a^n b^n$ (bottom), including the start/end-of-sequence symbol ‘#’. For $a^n b^n$, accuracy is measured at deterministic steps, after the first ‘b’. For Dyck-1, accuracy is the fraction of time steps where a model predicts only valid next symbols: ‘#’ should be predicted only when brackets are well balanced.

accuracy metric we add an adjustable error margin ε . Acceptance of a string is defined as reaching 100% accuracy (minus ε) on the string. Success on the test set is then defined as accepting all strings in the set (third condition in (1)).

1. *Deterministic accuracy* (M_{det}). Some languages contain strings with deterministic phases, where the next symbol is fully predictable. For example, strings in the language $a^n b^n$ have two phases, the a phase and the b phase. As long as only a ’s are seen, the next symbol remains unpredictable as the sequence can continue with another a or switch to the b phase. The string becomes deterministic once the first b appears. M_{det} is defined as the fraction of deterministic time steps in which the model assigns the majority probability to the correct next symbol. This metric is used in Joulin and Mikolov (2015).

A string is considered accepted if the model is $1 - \varepsilon$ accurate over all deterministic time steps. Note however that even a very small ε might benefit models that do not recognize strings well. For example, for the language $a^n b^n$, the deterministic steps in a string are the b ’s and the final end-of-sequence symbol. A degenerate model that predicts only b ’s will get only the end-of-sequence symbol wrong out of all deterministic steps, and will reach a very high accuracy score. For any large enough test set these errors will be hidden within the ε margin and the model will be deemed successful. ε should therefore be chosen with care per task.

M_{det} is used below for the following languages that have deterministic phases: $a^n b^n$, $a^n b^n c^n$, $a^n b^n c^n d^n$, and $a^n b^m c^{n+m}$.

2. *Categorical accuracy* (M_{cat}). Some language strings do not have any predictable phases. This is the case in the Dyck family of languages. At each time step in a Dyck string, one may open a new bracket (see Figure 2). M_{cat} is therefore defined as the fraction of steps in which a network assigns probability $p > \varepsilon$ to each possible next symbol, and $p \leq \varepsilon$ to irrelevant symbols. Non-probabilistic versions of M_{cat} are used in Gers and Schmidhuber (2001) and Suzgun et al. (2019a, 2019b) who do not treat network outputs as probability distributions. M_{cat} is used below for Dyck languages.

As specified in Section 3.1, the index \mathbb{B} is calculated based on the largest test set for which a model reaches an ε -perfect accuracy score.

Beyond accuracy, one might be interested in inspecting a model’s knowledge of the distribution of strings in \mathcal{L} induced by a probabilistic G . This can be done by using the probabilistic sampling method described in Section 3.4 and accompanying it with a probabilistic accuracy measure – for example, one based on an optimal cross-entropy score, which is known from G ’s expansion probabilities (as done in Lan et al., 2022). Feeding loss values into an accuracy metric will require normalizing them across tasks. We leave this extension for future work.

3.6 String structure

Following Gers and Schmidhuber (2001), each sequence starts and ends with a start/end-of-sequence symbol ‘#’. This turns the task into a strict acceptance/rejection task – predicting the end-of-sequence symbol is taken as going into an accept state. The start- and end-of-sequence symbols are added to the task-specific vocabulary and are assigned probabilities by the model at each step. Figure 2 illustrates input and target sequences for $a^n b^n$ and Dyck-1.

3.7 Limitations

One shortcoming of the proposed index score is that it does not reflect perfect generalization, i.e., it is an empirical index that cannot point out a model that outputs correct predictions for *any* string in $\mathcal{L}(G)$. For most models, this will not be a problem, and \mathbb{B} will simply represent the model’s best training vs. test size ratio. In the case of a model that reaches perfect generalization on any input, the index score will represent the critical training size that brings the model to this performance.

Assigning a generalization score to infinitely correct models will remain a problem for any empirical metric that assigns scores to models based on finite test values. An alternative to such empirical probes would be to analytically show that a model is correct (as done in Lan et al., 2022).

4 Datasets

We provide training and test datasets for a preliminary set of formal languages for evaluation using the \mathbb{B} index. The dataset includes the languages $a^n b^n$, $a^n b^n c^n$, $a^n b^n c^n d^n$, $a^n b^m c^{n+m}$,

Dyck-1, and Dyck-2. The source code, datasets, and specifications for the benchmark are available at <https://github.com/taucmpling/bliss>.

4.1 Training and test sets

Training sets for context-free languages are sampled from PCFGs as described in Section 3.4. The PCFGs are given in Appendix B. Training sets for context-sensitive languages are generated by sampling values for n from a geometric distribution.

Test sets are generated using the method described in Section 3.4: All test sets consist of an exhaustive list of strings ordered by length starting right after the longest string seen during training. Test sets for $a^n b^m c^{n+m}$ consist of the list of strings starting after the last seen pair of n, m , sorted by $n + m$ values to test all possible combinations.

5 Experiments

5.1 Models

We test the following models: LSTM RNNs (Hochreiter and Schmidhuber, 1997); Memory-augmented RNNs (MARNN; Suzgun et al., 2019a); and Minimum Description Length RNNs (MDL-RNN; Lan et al., 2022).

LSTM architectures were developed with the task of keeping items in memory over long distances in mind. As mentioned above, Weiss et al. (2018) have shown that LSTMs are theoretically capable of recognizing CL.

MARNNs (Suzgun et al., 2019b) are RNNs equipped with external memory devices, and were shown to yield better performance when learning languages that require stack-like devices and beyond. Here we use Stack-LSTM, an LSTM augmented with a pushdown automaton; and Baby Neural Turing Machines (Baby-NTM; itself a variant of NTMs, Graves et al., 2014), an RNN with a more freely manipulable memory.³

MDLRNNs are RNNs trained to optimize the Minimum Description Length objective (MDL; Rissanen, 1978), a computable approximation of Kolmogorov complexity, the algorithmic complexity of a model. The intuition behind the objective is equating compression with finding regularities in the data: a model that compresses the data well will generalize better and avoid overfitting. In practice, optimization is done by minimizing the sum of

³We modify Suzgun et al. (2019a)’s models to output probability distributions, replacing the final sigmoids with a softmax layer and the MSE loss with cross-entropy. See Section 3.3.

the architecture encoding length and the standard cross-entropy loss, both measured in bits based on a specific encoding scheme.

MDL is a stricter regularizer than standard regularization techniques such as L1/L2: the latter penalize large weight values but cannot prevent models from overfitting using a solution that uses many small, but informative, weights. MDL penalizes the actual information content of the network, forcing it to be general and avoid overfitting. MDLRNNs were shown to learn some of the languages discussed here in full generality using small architectures of only 1 or 2 hidden units and to outperform L1/L2 (Lan et al., 2022).

MDL is a non-differentiable objective, which requires that MDLRNN be optimized using a non-gradient based search method, such as an evolutionary algorithm that searches the network architecture space. Since this method is not based on gradient descent, Lan et al. (2022) were able to use non-standard, non-differentiable activations such as step functions. Here we restrict the architecture space to only standard activations: the linear function, ReLU, and tanh. This serves both to compare MDLRNN with standard networks and to limit the architecture search space. We publish the resulting nets as part of the MDLRNN-Torch release at <https://github.com/0xnurl/mdlrnn-torch>.

Appendix A lists the hyper-params for all runs.

5.2 Training sets

We used training sizes $|\mathcal{C}| = 100, 250, 500, 1000$. We stopped at the smallest size 100 because in our setup this size results in test strings of lengths $> 10,000$, leading to very long running times.

5.3 Index parameters

We calculate the \mathbb{B} index for all trained networks using the following index parameters:

Magnitude parameter $N = 3$, i.e., training and test sizes are derived from a baseline size 10^3 . This order of magnitude was selected based on the training set sizes used in previous works for the languages inspected here (Table 1).

$M_{det} \varepsilon = 0.005$, i.e., a model needs to correctly predict the next symbol for at least 99.5% of all deterministic steps. Since even this high threshold allows a degenerate model to reach good scores as described in Section 3.5, we also calculate the index score using $\varepsilon = 0$, i.e. a model must predict *all* deterministic symbols correctly.

Language	Model	\mathbb{B} -index	
		$\varepsilon = 0.005$	$\varepsilon = 0$
$a^n b^n$	LSTM	10	<1
	Stack-LSTM	10	<1
	Baby-NTM	10	1
	MDLRNN	10	10
$a^n b^n c^n$	LSTM	<1	<1
	Stack-LSTM	2	<1
	Baby-NTM	10	<1
	MDLRNN	<1	<1
$a^n b^n c^n d^n$	LSTM	<1	<1
	Stack-LSTM	1	<1
	Baby-NTM	4	<1
	MDLRNN	<1	<1
$a^n b^m c^{n+m}$	LSTM	<1	<1
	Stack-LSTM	10	<1
	Baby-NTM	4	<1
	MDLRNN	4	4
Dyck-1	LSTM	<1	<1
	Stack-LSTM	<1	<1
	Baby-NTM	<1	<1
	MDLRNN	2	2
Dyck-2	LSTM	<1	<1
	Stack-LSTM	<1	<1
	Baby-NTM	<1	<1
	MDLRNN	<1	<1

Table 2: Generalization scores \mathbb{B} . The index represents how well a model generalizes in relation to its training size. A score $\mathbb{B} = 4$ indicates that a model trained on 250 samples reached the accuracy criterion on the consecutive 4,000 unseen test samples. $\mathbb{B} < 1$ indicates that the model did not reach the accuracy criterion when the test size was greater than the training size, but might reach it for larger training and smaller test sets.

$M_{cat} \varepsilon = 0.005$, i.e., for Dyck, a model needs to assign $p \leq 0.005$ to each irrelevant symbol and $p > 0.005$ to possible ones. Here as well we report results for $\varepsilon = 0$, i.e., a model must assign non-zero probabilities to valid symbols only.

6 Results

6.1 Non-perfect accuracy

The generalization index obtained by each model for each language is presented in Table 2.

We start by inspecting the indexes calculated using the more lenient accuracy margin $\varepsilon = 0.005$.

For $a^n b^n$, under this accuracy margin, all models are assigned index $\mathbb{B} = 10$, i.e., reaching the success criterion for the next unseen 10,000 samples

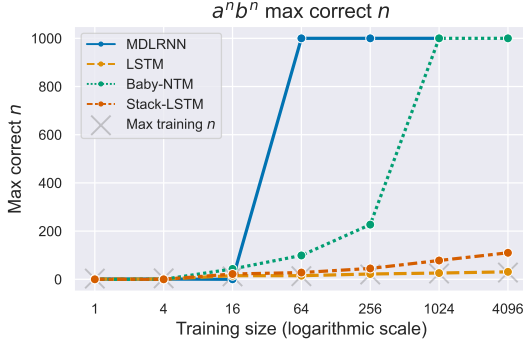


Figure 3: Generalization performance of the models tested here. Models were trained on strings drawn from $a^n b^n$ and tested on acceptance of strings up to $n = 1,000$. X’s mark the maximum n seen during training.

after being trained on 100 samples. For the specific combination of random seed and sampling prior in these experiments, this means that the models were trained on strings up to $a^{20}b^{20}$ and generalized to all strings up to at least $a^{10020}b^{10020}$ with deterministic accuracy $M_{det} \geq 99.5\%$.

For $a^n b^n c^n$, MARNNs reach $\mathbb{B} = 10$ and 2, while LSTM and MDLRNNs do not reach the success criterion, resulting in $\mathbb{B} < 1$. For $a^n b^n c^n d^n$ only MARNNs reach a specified index, with a Baby-NTM reaching $\mathbb{B} = 4$, indicating that it generalized to strings as long as $a^{4020}b^{4020}c^{4020}d^{4020}$ with $M_{det} \geq 99.5\%$.

For the addition language $a^n b^m c^{n+m}$, Stack-LSTM and MDLRNN reached index scores $\mathbb{B} = 10$ and 4 respectively. For the specific combination of random seed and the sampling prior used here, this means that the winning Stack-LSTM saw maximum values of $n = 18, m = 20$ during training, and generalized to all strings up to $a^{120}b^{120}c^{240}$ with $M_{det} \geq 99.5\%$.

6.2 Perfect accuracy

We report the generalization scores using a strict $\varepsilon = 0$ as well, i.e., when a model is required to predict *all* deterministic steps correctly or assign non-zero probability to valid symbols only. For languages with deterministic steps such as $a^n b^n$, this means that the model needs to always predict the end-of-sequence symbol correctly, thus making a distinction between accepting a string and approximating its surface structure.

Here, only MDLRNNs remain at the same scores, indicating that they predicted all time steps correctly. Baby-NTM reaches $\mathbb{B} = 1$ for $a^n b^n$, a

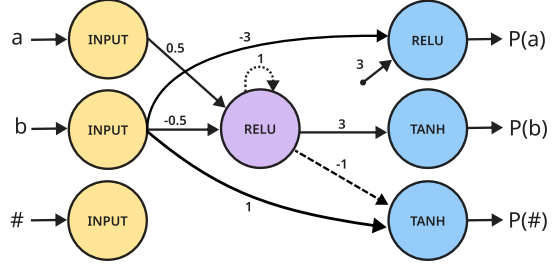


Figure 4: RNN cell architecture of the best-performing MDLRNN for $a^n b^n$, which trained on 100 samples and reached $\mathbb{B}_3 = 10$. The network uses only one hidden unit and standard activation functions, and generalizes up to at least $a^{35000}b^{35000}$. Dashed arrows are recurrent connections across time steps. The loop from the hidden ReLU unit to itself is a counter mechanism evolved by the evolutionary algorithm to count and compare the number of a ’s and b ’s.

drop from 10. The rest of the networks drop to $\mathbb{B} < 1$, revealing that their good scores in the previous comparison calculated with a non-zero ε was due to them approximating the target languages, even at low n values.

MDLRNN performance here is in line with results from Lan et al. (2022), who provided evidence that MDLRNNs for these languages do not only perform empirically well on large test values, but are also provably correct for any input. However, here we limited activations to standard, non-discrete functions (Section 5.1), potentially limiting the network’s ability to generalize well in the limit. While we do not provide correctness proofs for the networks found here, the index scores indicate that MDLRNNs generalize well to large values using only standard activations. Figure 4 presents the MDLRNN found for $a^n b^n$. We checked whether this network also accepts n values beyond those needed to reach the score $\mathbb{B} = 10$ ($n = 10,020$). The network reached 100% M_{det} for all values up to $n = 35,000$, at which point we stopped the test due to long feeding times.

Beyond the benchmark scores, Figure 3 plots the largest n value for $a^n b^n$ strings predicted by the models tested here with 100% M_{det} accuracy ($\varepsilon = 0$), as a function of training set size. Both MDLRNNs and Baby-NTMs reach perfect accuracy up to the tested maximum of $n = 1,000$. MDLRNNs however require two orders of magnitude less data to reach this performance (and the benchmark scores in Table 2 show that in fact MDLRNNs generalized up to at least $n = 10,000$, while Baby-NTMs remained at 1,000). LSTMs

and Stack-RNNs did not generalize well beyond the training samples. This is in line with previous works showing that these models may need substantially more training data in order to learn these languages (Table 1).

7 Discussion

We provided a simple index for how well a model generalizes: how much it can learn from how little data. We illustrated the usefulness of this index in a comparison of several current models over several formal languages. Beyond showing which current models generalize better than others, the benchmark also highlights which aspects of artificial neural networks work well for grammar induction, and what is still missing.

Among languages that were learned with perfect accuracy ($a^n b^n$, $a^n b^m c^{n+m}$, Dyck-1), MDLRNNs generalized best, but still failed on others ($a^n b^n c^n$, $a^n b^n c^n d^n$, and Dyck-2). Previous work has shown that this model’s search procedure, an evolutionary algorithm, fails to find networks that are manually shown to have better MDL scores (Lan et al., 2022). We take this to show that the optimization procedure limits the model and prevents it from taking full advantage of the MDL objective. The benefit of the MDL objective is nevertheless evident in the generalization performance for several languages.

MARNNs clearly benefit from their memory devices and reach good generalization scores, but testing for perfect accuracy ($\varepsilon = 0$) reveals that their learning outcome is mostly approximate, and that they fail to maintain perfect accuracy for long stretches beyond their training data. This could be the result of an inadequate objective function (cross-entropy), limitations of the search (backpropagation/gradient descent), or both. We do not currently have results that help decide this matter, but recent results for other architectures (El-Naggar et al., 2023b) hint that the problem lies at least in part in the objective function.

8 Acknowledgements

This work was granted access to the HPC resources of IDRIS under the allocation 2023-AD011013783 made by GENCI.

References

Mikael Bodén and Janet Wiles. 2000. Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Science*, 12(3-4):197–210.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. 2022. [Neural Networks and the Chomsky Hierarchy](#).

Nadine El-Naggar, Pranava Madhyastha, and Tillman Weyde. 2022. [Exploring the Long-Term Generalization of Counting Behavior in RNNs](#).

Nadine El-Naggar, Pranava Madhyastha, and Tillman Weyde. 2023a. [Theoretical Conditions and Empirical Failure of Bracket Counting on Long Sequences with Linear Recurrent Networks](#).

Nadine El-Naggar, Andrew Ryzhikov, Laure Daviaud, Pranava Madhyastha, and Tillman Weyde. 2023b. Formal and empirical studies of counting behaviour in ReLU RNNs. In *Proceedings of 16th Edition of the International Conference on Grammatical Inference*, volume 217 of *Proceedings of Machine Learning Research*, pages 199–222. PMLR.

W. Tecumseh Fitch and Marc D. Hauser. 2004. Computational constraints on syntactic processing in a nonhuman primate. *Science*, 303(5656):377–380.

Felix Gers and Jürgen Schmidhuber. 2001. [LSTM recurrent networks learn simple context-free and context-sensitive languages](#). *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. [Neural Turing Machines](#). *arXiv:1410.5401 [cs]*.

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

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. 2023. [Stop Uploading Test Data in Plain Text: Practical Strategies for Mitigating Data Contamination by Evaluation Benchmarks](#).

Armand Joulin and Tomas Mikolov. 2015. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Diederik P. Kingma and Jimmy Ba. 2017. [Adam: A Method for Stochastic Optimization](#).

Nur Lan, Michal Geyer, Emmanuel Chemla, and Roni Katzir. 2022. [Minimum Description Length Recurrent Neural Networks](#). *Transactions of the Association for Computational Linguistics*, 10:785–799.

Raphaëlle Malassis, Stanislas Dehaene, and Joël Fagot. 2020. [Baboons \(Papio papio\) Process a Context-Free but Not a Context-Sensitive Grammar](#). *Scientific Reports*, 10(1):7381.

William Merrill. 2021. [On the Linguistic Capacity of Real-Time Counter Automata](#).

J. Rissanen. 1978. [Modeling by shortest data description](#). *Automatica*, 14(5):465–471.

Hava T. Siegelmann and Eduardo D. Sontag. 1992. [On the computational power of neural nets](#). In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, pages 440–449, New York, NY, USA. Association for Computing Machinery.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019a. [LSTM Networks Can Perform Dynamic Counting](#). In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.

Mirac Suzgun, Sebastian Gehrmann, Yonatan Belinkov, and Stuart M. Shieber. 2019b. [Memory-Augmented Recurrent Neural Networks Can Learn Generalized Dyck Languages](#). *arXiv:1911.03329 [cs]*.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745.

A Appendix: Hyper-parameters

A.1 Training corpora

All training sets were generated using the same random seed 100 and prior probability $p = 0.3$. The datasets are available at <https://github.com/taucompling/bliss>. Following [Jacovi et al. \(2023\)](#), the datasets are zipped and password-protected to prevent test data contamination of large language models through crawling.

Each of the LSTM and MARNN hyper-param configurations below was run 3 times using different random seeds (100, 101, 102). MDLRRNs were run once per configuration because of their longer running time.

A.2 LSTM

LSTMs were trained based on the following hyper-params grid: hidden state size (2/32/128), regularization technique (L1/L2/none), and the regularization constant in case regularization was applied ($\lambda = 1.0/0.1/0.01$). Networks were trained using the Adam optimizer ([Kingma and Ba, 2017](#)) with learning rate 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The networks were trained by feeding the full batch of training data for 1,000 epochs.

A.3 MARNN

MARNNs were trained by varying the architecture type (Softmax Stack-LSTM/Softmax Baby-NTM) and stack/memory size (50/100 for Stack-LSTM, 2050 for Baby-NTM). For Stack-LSTM,

stack sizes were selected so they were always larger than the largest values seen during training: $n + m = 22 + 24$ for $a^n b^m c^{n+m}$ and $n = 24$ for all other languages. During testing the stack size was enlarged to 2050, beyond the maximum needed to reach scores $\mathbb{B} = 1$ and 2. Baby-NTM memory was set to 2050 already during training because this model’s memory size affects the weight dimensions and cannot be changed after training.

The rest of the hyper-parameters were set to the default values from [Suzgun et al. \(2019b\)](#). Stack-LSTM: hidden size 8; 1 layer; memory dimension 5; epochs 3/50; learning rate 0.01; Baby-NTM: hidden size 8; 1 layer; memory dimension 5; epochs 3/50; learning rate 0.01.

The original MARNN setup was modified here so that the network outputs represent probability distributions and not multi-label outputs. This was done by replacing the sigmoid outputs with a softmax layer and the MSE loss with cross-entropy.

A.4 MDLRRN

MDLRRNs were trained using the evolutionary algorithm and the same hyper-params reported in [Lan et al. \(2022\)](#): population size 500; islands size 250; 25,000 generations; tournament size 2; early stop after 2 hours of no improvement; elite ratio 0.001; migration interval 1,000 generations/30 minutes.

B Appendix: PCFGs

B.1 $a^n b^n$

$$S \rightarrow \begin{cases} aSb & 1-p \\ \varepsilon & p \end{cases}$$

B.2 $a^n b^m c^{n+m}$

$$S \rightarrow \begin{cases} aSc & 1-p \\ X & p \end{cases}$$

$$X \rightarrow \begin{cases} bXc & 1-p \\ \varepsilon & p \end{cases}$$

B.3 Dyck-1

$$S \rightarrow \begin{cases} (S)S & p \\ \varepsilon & 1-p \end{cases}$$

B.4 Dyck-2

$$S \rightarrow \begin{cases} (S)S & p/2 \\ [S]S & p/2 \\ \varepsilon & 1-p \end{cases}$$

Chapter 3

Bridging the Empirical-Theoretical Gap in Neural Network Formal Language Learning Using Minimum Description Length (joint with Emmanuel Chemla and Roni Katzir)

Nur Lan, Emmanuel Chemla, and Roni Katzir. 2024. Bridging the Empirical-Theoretical Gap in Neural Network Formal Language Learning Using Minimum Description Length (arXiv:2402.10013). *arXiv*. <https://doi.org/10.48550/arXiv.2402.10013>

Bridging the Empirical-Theoretical Gap in Neural Network Formal Language Learning Using Minimum Description Length

Nur Lan^{1,2}, Emmanuel Chemla^{1,3}, Roni Katzir²

¹Ecole Normale Supérieure, ²Tel Aviv University, ³EHESS, PSL University, CNRS
{nur.lan, emmanuel.chemla}@ens.psl.eu
rkatzir@tauex.tau.ac.il

Abstract

Neural networks offer good approximation to many tasks but consistently fail to reach perfect generalization, even when theoretical work shows that such perfect solutions can be expressed by certain architectures. Using the task of formal language learning, we focus on one simple formal language and show that the theoretically correct solution is in fact not an optimum of commonly used objectives — even with regularization techniques that according to common wisdom should lead to simple weights and good generalization (L1, L2) or other meta-heuristics (early-stopping, dropout). However, replacing standard targets with the Minimum Description Length objective (MDL) results in the correct solution being an optimum.

1 Introduction

Probing Artificial Neural Networks’ (ANNs) capabilities in the domain of language learning has advanced in two complementary paths – theoretical and empirical. Theoretical work tries to delineate the kinds of languages and phenomena that can be expressed by ANNs, and empirical work involves training networks on such tasks and inspecting their performance.

An often overlooked fact is that these paths have still not converged: while theoretical work continues to provide findings regarding the expressivity of different architectures, empirical work keeps arriving at suboptimal solutions that fall short of the theoretically correct ones. For example, for formal languages such as $a^n b^n$ or Dyck-1, among many others, we are not aware of any network trained through gradient descent that was shown to perform well on strings that are orders of magnitudes longer than those seen during training, while failures at low lengths are pervasive (Joulin and Mikolov, 2015, Weiss et al., 2018, Suzgun et al., 2019, Bhattamishra et al., 2020, El-Naggar et al.,

2022, among others; see Lan et al., 2023 for an overview). This stands in contrast to symbolic models, where the requirement for solution correctness across lengths is trivially met.

Why this would be the case is often either left unexplained or waved off as a shortcoming of the optimization method (most often, gradient descent using backpropagation). In this work we argue that these failures are not due to training misfortunes that could be overcome, for example, by using a more exhaustive hyper-parameter search. Rather, they are due to inherent characteristics of the training objectives currently used for such tasks.

Our main contributions are:

1. We present a manually built, optimal Long Short-Term Memory network (LSTM; Hochreiter and Schmidhuber, 1997) that accepts the formal language $a^n b^n$, following a general recipe given in Weiss et al. (2018). We show that this network would not be found using standard training objectives, since it does not lie at optimum points of these objectives – even when using regularization terms which according to common wisdom should result in general solutions.
2. We show that by replacing these objectives and regularization terms with an objective to minimize the network’s Minimum Description Length (MDL, Rissanen, 1978), accompanied by an intuitive encoding scheme, the optimal network becomes an optimum of the objective.

The full experimental materials and source code are available at <https://github.com/0xnurl/mdl-lstm>.

2 Previous work

We rely mainly on three recent works, which we extend in the following ways. First, the current

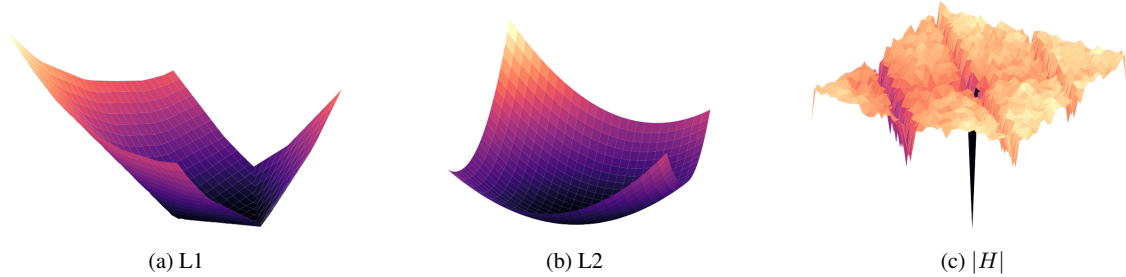


Figure 1: Loss surfaces around the golden $a^n b^n$ LSTM from Section 4, for the regularization terms considered here: L1, L2, and $|H|$ – the hypothesis encoding length term of the MDL objective. $|H|$ is jagged and non-differentiable but results in the correct net being an optimum of the full loss function (Figure 4).

work is similar to [El-Naggar et al. \(2023\)](#), who inspected the role of the objective function in formal language learning. They showed that for a simple recurrent neural network (RNN), which uses a single ReLU layer, the optimal counting solution does not align with optima of common loss functions (cross-entropy and mean squared error). This was done by providing necessary and sufficient conditions for implementing counting in a ReLU-RNN. We extend this work in the following ways. First, we move to the more commonly used LSTM RNN. Since this architecture is more complex, it is also harder to find such sufficient and necessary conditions for counting as done by [El-Naggar et al. \(2023\)](#). This leaves our results mostly empirical, compared to their analytical result. However, we go beyond that work by also providing an alternative objective (MDL), for which the optimal network becomes an optimum.

Second, in order to locate such an optimum of the objective, we build an optimal LSTM that accepts a specific formal language. For this we rely on [Weiss et al. \(2018\)](#), who showed that an LSTM can theoretically implement counting using specific weight configurations, so that the state vector holds a counter that can be incremented and decremented based on the input. Here we implement their general recipe to build an optimum LSTM that accepts the language $a^n b^n$. We focus on one language for simplicity, and the method can be easily extended to more languages.

Third and closest to the current work, [Lan et al. \(2022\)](#) applied the MDL principle to RNNs for formal language learning. The resulting networks were shown to be correct for any string for languages such as $a^n b^n$, $a^n b^m c^{n+m}$, and Dyck-1. Since this objective resulted in a non-differentiable loss function, [Lan et al. \(2022\)](#) used neuroevolution

to search the hypothesis space, evolving free-form RNN cells. Since our focus in this work is the objective, here we leave the search algorithm aside and use a single fixed architecture for which a theoretical target is known to exist (LSTM). We then inspect the effect of the objective function on potential weight solutions.

More broadly, empirical work using RNNs for artificial grammar learning have been carried out at least since the introduction of Simple RNNs in [Elman \(1990\)](#). Theoretical work regarding RNNs’ theoretical computational power go back at least to [Siegelmann and Sontag \(1992\)](#), who showed that RNNs are Turing-complete under certain permissive assumptions (infinite activation precision and unbounded running time). The empirical success of ANNs in the practical field of natural language processing (NLP) has led to recent interest in the theoretical power of RNNs under practical conditions, mainly real-time processing and finite precision ([Weiss et al., 2018](#), [Merrill et al., 2020](#)). Other recent work has applied similar methods to the transformer architecture (see survey in [Strobl et al., 2023](#)).

In terms of empirical results, works since [Elman \(1990\)](#) trained ANNs to recognize formal languages and most often tested for generalization using unseen string lengths and depths ([Gers and Schmidhuber, 2001](#), [Joulin and Mikolov, 2015](#), among many others). [Lan et al. \(2023\)](#) provide an overview of such works; they show that common to these works is a failure to generalize beyond a certain tested length. [Lan et al. \(2023\)](#) also provide a standardized benchmark for formal language learning, and find that RNNs trained to optimize standard losses fail to generalize well from reasonably small amounts of data, while an RNN variant trained to minimize MDL ([Lan et al., 2022](#)) is

able to generalize significantly better (potentially infinitely so).

Applying the MDL criterion to ANNs also dates back to at least the early 1990’s. (See [Schmidhuber, 1997](#) for an overview of early attempts in this area, and [Lan et al., 2022](#) for a review of more recent work.) [Hinton and Van Camp \(1993\)](#) minimized the encoding length of the weights alongside the prediction error, while leaving the architecture fixed. [Hochreiter and Schmidhuber \(1994\)](#) provided an algorithm that searches for networks that lie at ‘flat minima’ – regions of parameter space where error remains relatively similar; this preference is given an MDL justification. [Zhang and Muhlenbein \(1993\)](#) used a genetic algorithm to search for network architectures that minimize an MDL score, using a weight encoding similar to L2 regularization. [Schmidhuber \(1997\)](#) presented an algorithm for discovering networks that optimize a running-time based complexity metric that is closely related to MDL (Levin complexity).

3 General setup

We describe here the technical background leading to the experiments in Section 4.

3.1 Minimum Description Length

Striking a balance between model complexity and its fit to the data is important in order to avoid both overfitting and underfitting. It is generally assumed that minimizing model complexity is good (Occam’s razor). This general principle was formalized within Kolmogorov Complexity (KC; [Solomonoff, 1964](#), [Chaitin, 1966](#), [Kolmogorov, 1968](#)), defined as the length of the shortest program that generates specific data.

KC however is non-computable, a result of the target representation being Turing-complete. The Minimum Description Length principle (MDL; [Rissanen, 1978](#)) makes it possible to escape the non-computability of KC by relaxing the requirement of a Turing-complete representation, and moving to a less powerful formalism (for example, a context-free grammar).

Formally, consider a hypothesis space \mathcal{H} and input data D . The MDL principle aims to find a hypothesis H^* that minimizes the sum:

$$H^* = \arg \min_{H \in \mathcal{H}} |H|_C + |D : H| \quad (1)$$

where $|H|_C$ is the length of H encoded using an encoding scheme C for encoding hypotheses in

\mathcal{H} . Encoding length is usually measured in bits. $|D : H|$ is the encoding length of D given H .

Minimizing $|H|_C$ alone would result in a degenerate, over-general model that does not fit the data well. Conversely, minimizing $|D : H|$ alone would result in overfitting. Minimizing both terms together results in a reasonable compromise between generalization and accuracy.

3.2 Encoding a network

In this work, hypotheses in \mathcal{H} are taken to be LSTM networks with one linear output layer, followed by a softmax function. Here we describe an encoding scheme for such networks which makes it possible to measure their encoding length $|H|$.

We first note that a reasonable encoding scheme for networks should follow an intuitive notion of simplicity in order to penalize overfitting (i.e., lead to larger encoding length). Equating scalar magnitude with simplicity is not enough, since it is still possible to ‘smuggle’ large amounts of information inside very small scalar values. One extreme example is using a fractal encoding in the spirit of [Siegelmann and Sontag \(1992\)](#) or [Tabor \(2000\)](#) which encodes a stack inside a small rational number.^{1,2} However, less sophisticated overfitting is also conceivable using highly specific weights, for example if a model assigns specific probabilities due to sampling artefacts in the training set. A reasonable objective should make such memorization worthwhile only if the data justify it, e.g., if it contains many repetitions of the same pattern.

Regularization terms such as L1/L2 are not good enough then, since they would deem, for example, a simple value such as 1 worse than a smaller yet highly informative value (e.g., $\sum_1^n \frac{2w_i+1}{4^i} < 1$, the fractal encoding of a binary stack w , from [Siegelmann and Sontag, 1992](#)).³

For MDL, the encoding scheme C explained in Section 3.1 needs to be chosen so that it fulfill the simplicity requirement. We opt for the following encoding scheme, used in [Lan et al. \(2022\)](#).

¹These works use activation values, not weights, to store such values. However, such a construction still illustrates the difference between information content and scalar magnitude.

²Admittedly and as discussed also in [Weiss et al. \(2018\)](#), standard gradient-based methods would most probably not reach such highly specific weight configurations. This does not mean however that such solutions do not exist in the search space, and that better search algorithms would not find them.

³Note that simply taking the log of the weight’s value, thus roughly converting it to its length in bits, would not be enough either: this would simply result in a rescaling of the weight.

A weight w_{ij} is represented as a rational fraction $\frac{n}{m}$. The numerator and denominator are encoded using the prefix-free encoding for integers from Li and Vitányi (2008):

$$E(n) = \underbrace{11111 \dots 1111}_{\text{Unary enc. of } \lceil \log_2 n \rceil} \underbrace{0}_{\text{Separator}} \underbrace{10101 \dots 00110}_{\text{Binary enc. of } n}$$

Both encodings are then concatenated, with an extra bit for the sign. For example, the weight $w_{ij} = +\frac{2}{5}$ would be encoded as:

$$\underbrace{\begin{array}{c} 1 \\ + \end{array}} \underbrace{E(2) = 11010}_{2} \underbrace{E(5) = 1110101}_{5} \\ w_{ij}$$

This encoding fulfills the requirement above: the encoding of very specific or informative weights would be considerably longer than that of intuitively simpler values such as 1.

In the current setup, the LSTM architectures vary only by the size of the hidden vector and the values of the weights. In order to reliably encode a specific network one needs to encode only the weights of the LSTM cell and output layer, and prepend the size of the hidden vector. The encoding of a specific network would then be:

$$\underbrace{\begin{array}{c} \underbrace{11011}_{E(\text{hidden size})} \quad \underbrace{11 \dots 01}_{\dots} \quad \underbrace{10 \dots 01}_{w_{ij}} \quad \underbrace{11 \dots 10}_{\dots} \\ \text{Weight encoding} \end{array}}_{\text{LSTM encoding}}$$

To calculate $|H|$ for networks trained through backpropagation with floating-point weights, in sections below floats are converted to the closest rational with denominator $m \leq 1000$.

3.3 Language modeling

We use the formal language $a^n b^n$ as a test case throughout this work, and probe different networks' performance on recognizing it. Strings are drawn from the following probabilistic context-free grammar (PCFG):

$$S \rightarrow \begin{cases} aSb & 1-p \\ \varepsilon & p \end{cases} \quad (2)$$

with $p = 0.3$ for all tasks. We use a standard language modeling setup in which the network is fed one symbol at a time, and outputs a probability distribution over the alphabet, predicting the next

symbol in the string. Following Gers and Schmidhuber (2001), each string starts and ends with a special symbol.

The training set is sampled by generating strings from (2). The validation set consists of all consecutive strings starting right after the last n in the training set. The validation loss is weighted per-sample so that it follows the same power law distribution induced by (2). The train-validation split in all experiments is 95%-5%. In the following sections the training size is 1,000, i.e., a 950-50 split. The maximum n in this training set was 21, so the validation set contained all strings with $22 \leq n \leq 71$. The test set in all experiments consisted of all $a^n b^n$ strings with $1 \leq n \leq 1,500$.

The network is fed one symbol at a time, and at each step outputs a probability distribution \hat{p} over the alphabet for predicting the next symbol in the string. The baseline loss function we use is the standard cross-entropy loss (CE):

$$CE(p, \hat{p}) = - \sum_{i=0}^n p(c_i) \log(\hat{p}(c_i)) \quad (3)$$

where n is the length of a sequence, c_i is the target symbol at time step i , $p(c_i)$ is the target probability at time step i for this symbol, and $\hat{p}(c_i)$ is the probability assigned by the network to this symbol at this time step. In a language modeling setting the target $p(c_i)$ is set to 1, resulting in:

$$CE(p, \hat{p}) = - \sum_{i=0}^n \log(\hat{p}(c_i)) \quad (4)$$

This sum is then averaged over all time steps for all sequences.

To measure accuracy on the task, we use *deterministic accuracy* (Joulin and Mikolov, 2015, Lan et al., 2023), defined as the ratio of correct answers at parts of the string that are completely predictable (a correct answer being the network assigning the maximum probability to the correct next symbol). For $a^n b^n$ strings, this means measuring accuracy at the phase that starts once the first 'b' appears, including the end-of-sequence symbol. Measuring accuracy at the end-of-sequence symbol turns the task into a strict acceptance task and can distinguish a good network that correctly balances the number of a's and b's, from a degenerate network that, e.g. gets a high deterministic accuracy score simply by only predicting b's.

3.4 Loss surface exploration

Our goal is to test which objectives could lead to optimal solutions. While exhaustive search of the parameter space is infeasible, we can explore only parts of the loss space and check if an objective function turns out to favor suboptimal solutions over an optimal one. This would be an indicator that this objective is not suitable for the task (and would lead to reliance on meta-heuristics such as early stopping). We do this by exploring the loss surfaces around an optimal network that solves the task perfectly and around a backpropagation-trained network.

For a given network with parameters θ , and for a loss function L , the network’s loss is $L(\theta)$ (for some input x , omitted here). For the 2D visualization we use below, the area around a specific network’s θ can then be explored by using two direction vectors δ and η , and plotting:

$$f(\alpha, \beta) = L(\theta + \alpha\delta + \beta\eta) \quad (5)$$

We use the exploration technique by Li et al. (2018): δ and η are randomized from a Gaussian; then, specific parts of each direction vector are normalized so that they have the norm of the respective parts in the original θ . For fully-connected layers like the ones used in LSTMs, normalization is done for each set of weights leading to a specific neuron. This normalization technique preserves the relative scale of different weight components of a network, and was shown to better reflect properties like convexity when exploring a network’s surrounding space. In all plots below we use 51 equally spaced values of $\alpha, \beta \in [-1, 1]$. Exploration using larger ranges did not affect the results either visually or quantitatively.

3.5 Objectives

The objective functions for all tasks below share the following structure:

$$L(\theta) = CE + \lambda Reg(\theta),$$

Here, CE is the training cross-entropy loss (4) using the distribution \hat{p} outputted by the network. For the MDL objective in (1), CE serves as $|D : H|$. This can be justified in encoding-length terms since (4) gives the expected length in bits for transmitting the string using Shannon-Fano encoding.

In the second term, Reg is either $L_1(\theta) = \sum_{w_{ij} \in \theta} |w_{ij}|$, $L_2(\theta) = \sum_{w_{ij} \in \theta} w_{ij}^2$, or no regularization. For the MDL objective, $Reg(\theta)$ is $|H|$

– the encoding length of a network encoded using the method in Section 3.2. λ is a coefficient used to calibrate the level of regularization during training, and is usually chosen empirically.

The common wisdom motivating the regularization term in all cases is to prevent models from overfitting. In the L1/L2 regularization framework, this is done by preventing large weights. Using L1 also leads to a preference for zero-value weights, thus potentially removing connections altogether; this can be thought of as a differentiable way to perform architecture search. However, as suggested in Section 3.2, both terms do not seem like good proxies for the $|H|$ term, since small weights can in fact be very informative (i.e., very complex).

Figure 1 plots the three regularization terms considered here, surrounding the optimal network presented in Section 4. It can be seen that while the loss surfaces for L1/L2 are smooth, moving to MDL would result in a highly irregular surface, hostile to gradient methods. We return to the question of searching through this space in Section 6.

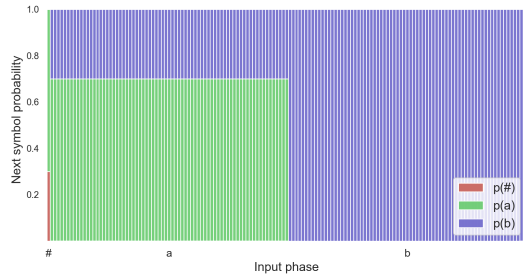
4 Optimal vs. trained $a^n b^n$ LSTM

Here we compare an optimal, manually-constructed LSTM that recognizes the language $a^n b^n$ perfectly, with an LSTM trained through backpropagation on the same task. We name the optimal network ‘golden’ to avoid confusion with general optimum points. The golden network is optimal in the sense that it always outputs the correct probabilities at each step of an $a^n b^n$ string drawn from (2), for any value of n . The optimal probabilities are presented in Figure 2a.

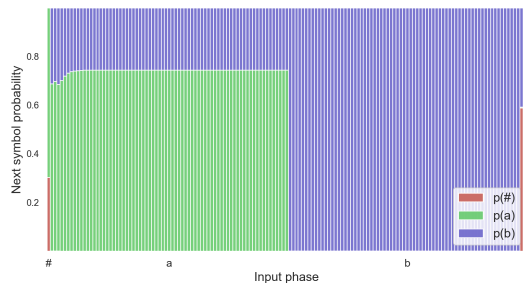
The goal of the experiment is to test whether a perfect solution can be found when using the different objectives considered here. Exhausting the entire parameter space is infeasible, even for networks with the very small hidden size (3) used here. However, if the golden network turns out to not be an optimum of certain objectives, i.e., a worse-performing network will be deemed better by an objective, we can conclude that this objective would not lead to this specific network.

4.1 Golden $a^n b^n$ network

The golden network is implemented based on a general recipe given in Weiss et al. (2018), who showed that an LSTM can theoretically implement counting using specific configurations of the gate weights, so that the state vector holds a counter



(a) Golden network



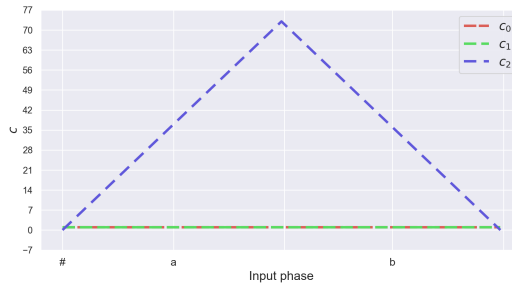
(b) Best trained network

Figure 2: Output probabilities assigned by the golden network and the best trained network for $n = 73$, the first point of failure of the trained network. Going left to right, each column represents the probability distribution outputted by the network at each time step.

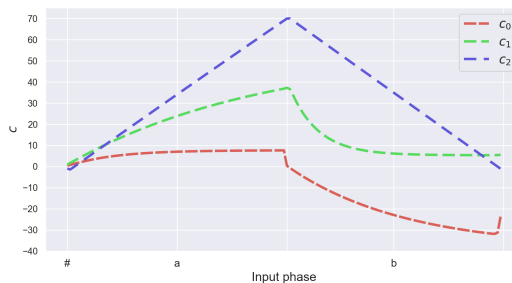
that can be incremented and decremented based on the input. This makes it possible for LSTMs to recognize a family of formal languages called Counter Languages. Roughly, this family corresponds to languages which can be recognized using a counting mechanism in real-time (Merrill, 2021). This includes $a^n b^n$. In an empirical experiment, however, Weiss et al. (2018) trained LSTMs on recognizing the language and found that the networks did not converge on the fully general counting solution, rather it converged on a suboptimal solution that failed to recognize $a^n b^n$ strings starting at n as low as 256.

We describe here in general terms the mechanics of the golden network. The full construction is given in Appendix B. The weights of the LSTM cell are set so that the network keeps track of the number of a 's compared to the number of b 's seen at each time step. Figure 3a plots the values of the memory vector of the LSTM during feeding of an $a^n b^n$ string, illustrating its counting mechanism. On top of the LSTM cell we add a linear layer that receives the hidden state as input, and outputs the correct target probabilities through a final softmax.

The manual network reaches 100% test accuracy



(a) Golden network



(b) Best trained network

Figure 3: Memory values for the golden network and the best trained network for $n = 73$, the first point of failure of the trained network. Each line corresponds to one component of c , the memory vector in the LSTM cell, as the string is fed to the network.

on the test set which contains all $a^n b^n$ strings with $1 \leq n \leq 1500$, and in fact can be shown to be correct for any n .

4.2 Backpropagation-trained $a^n b^n$ LSTM

We compare the golden network with networks trained on the same task using standard techniques. We run a hyper-parameter grid search to train LSTMs that have the same architecture as the manual network: hidden size 3 and a single linear output layer, followed by a softmax. The grid covers the following hyper-parameters: training set size, weight initialization method, regularization term, dropout rate, and early stopping patience based on validation loss, across five different random seeds. The grid yields 3,360 configurations. The full hyper-parameter grid is given in Appendix A.

The overfitting prevention techniques explored here belong to two groups: techniques where a regularization term is added directly to the loss function (L1, L2), and meta-heuristics external to the objective (dropout, early-stopping based on validation loss), aimed at preventing the loss from getting too low. Since our focus here is the objective function, we could potentially not include the

meta-heuristics. We still include dropout and early stopping however in order to compare the golden network with a network trained in a practical setting with the best chances to succeed. In Table 1 in the appendix we provide the same comparison for the best network that was trained without early stopping or dropout.

We select the best network out of all runs based on validation loss. The best network reached 77.3% test accuracy (i.e. on $1 \leq n \leq 1,500$) and was trained with the following parameters: training size 1000 (950-50 train-validation split); no regularization term; early stopping patience of 2 epochs; no dropout; normal weight initialization.

4.3 Network behavior

We start by comparing the behavior of the golden and trained networks. The best trained network correctly accepts all strings with $n \leq 72$ (the largest n in the training set was 21). We compare the networks using the first point of failure of the trained network, $n = 73$. Figure 2 plots the output probabilities assigned by the two networks throughout the sequence, and Figure 3 plots their memory values (c in the LSTM cell).

We first examine the network outputs in Figure 2. At first blush, the trained network seems successful, following the language distribution induced by (2) and visualized in Figure 2a almost perfectly. However, the network is imperfect in two ways: first, probabilities at the beginning of the string are incorrect, most probably due to overfitting of more frequent low- n values in the training set. Additionally and more crucially, the network’s count seems to leak, with probability mass for the end-of-sequence symbol assigned to the before-last time step. This becomes a problem for $n \geq 73$, when the network starts accepting illicit $a^n b^{n-1}$ strings.

As for the inner workings of the network, visualizing the network’s memory in Figure 3b shows that the network has indeed developed some counting mechanism in at least one component of the memory vector (it is unclear how it uses the other two), which seems to be imperfect as it goes below 0 towards the end of string.

4.4 Loss exploration

Is the suboptimal performance of the trained network above simply a misfortune of the current setup? We explore the possibility that the culprit might be the objective function. We do this by comparing the loss values of the golden network with

the trained network’s, using standard objectives and the MDL objective.

Beyond measuring the loss value of the two networks considered here, we also explore their surrounding loss landscape in order to check for alternative local minima and inspect properties like convexity and smoothness of the loss. This is done using the technique described in Section 3.4.

Figures 4 and 5 plot the different loss surfaces around the networks. On each plot we mark the minimum point in the neighborhood, to check if it aligns with the network under investigation. If it does not, using that loss (either for fine-tuning the network or training from scratch) would potentially end up at that other minimum. For each relevant point we use the parameter vector to build the corresponding LSTM, and calculate its test accuracy.

We start by exploring the loss surface around the golden network.⁴ Figures 4a and 4b show that if L1 or L2 regularization were used, the golden network would not have been found – rather, using these regularization terms would lead the search to suboptimal networks that have better training loss values, but also worse test accuracy. For the MDL loss, plotted in Figure 4c, the minimum aligns with the golden network, showing that at least in this neighborhood, searching with MDL as an objective would lead to the correct solution. In Section 6 we discuss potential limitations to these findings.

Figure 5 plots the different loss surfaces around the best trained network. We plot in 3D for comparison with the relevant value for the golden network, which lies in a different area of the loss space. Here, for all objectives, the winning network lies in a smooth and convex area. This is an expected result of using a gradient-based optimization. When evaluated using L1 and L2 regularization, the golden network ranks worse by the relevant losses. For the MDL objective the image is reversed: the trained network ranks worse, lying in the smooth basin seen in Figure 5c, while the MDL score of the golden network remains unreachable below. Since the two networks’ cross-entropy terms are almost identical (see Table 1), this inversion is mainly due to the $|H|$ term, which suggests that the trained

⁴We omit plotting the standalone cross-entropy loss because it is trivial to show that minimizing this loss alone will lead to overfitting (partially explaining the fact that the best performing network ends up using early stopping). Table 1 in the appendix demonstrates this using the next-best grid-search network that was trained without a regularization term or early stopping, whose cross-entropy loss goes below that of the golden network’s.

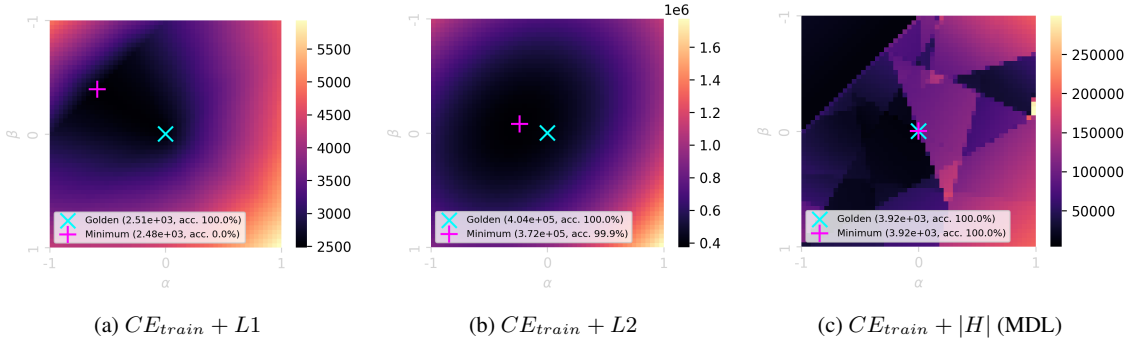


Figure 4: Training loss around the golden $a^n b^n$ LSTM, and test accuracy scores for the golden network and the local minimum network. Optimizing using L1 or L2 (4a, 4b) would result in suboptimal networks, while MDL (4c) results in alignment of the golden network with an optimum point of the loss.

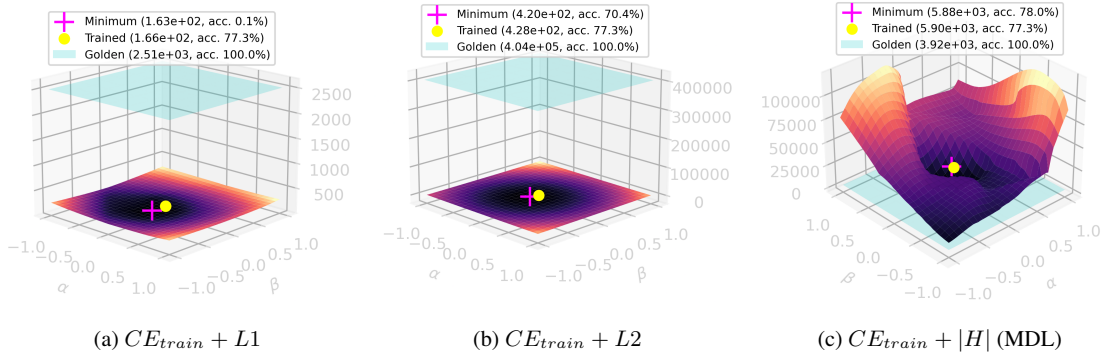


Figure 5: Training loss surfaces and the test accuracy of the best $a^n b^n$ LSTM found through a hyper-param grid search, trained using backpropagation with the standard cross-entropy loss and early stopping based on validation loss. When evaluated using L1 or L2 (5a, 5b) the network ranks better than the golden network, but has worse test performance. When evaluated using MDL (5c) the trained network does not minimize the loss as well as the golden network, ending up in a smooth but suboptimal area of the loss space.

network uses over-informative weights. Results for more λ values for all networks are given in Table 1.

5 Discussion

We presented a comparison between common overfitting-prevention techniques, among them some that equate simplicity with scalar magnitude, and the MDL objective accompanied with an encoding scheme for weights which favors an intuitive notion of simplicity. Combined with a manually built LSTM that optimally recognizes $a^n b^n$, we could measure the loss values of an optimal solution and check if they align with optimum points of the loss function. It was only when we used MDL that the optimal network aligned with the minimum of the loss. For the other loss functions, networks lying at minimum points had far from optimal performance. Using meta-heuristics such as early stopping mitigated overfitting to some extent,

but still did not lead to a fully general solution.

We interpret these findings as an indicator that ANNs’ failure to converge on provably existing, optimal solutions is not accidental, but rather an inherent and pathological property of the way that current models are trained. This is in line with a mounting list of generalization failures to learn formal languages, as well as more complicated natural language tasks.

We focused here on RNNs, mainly because they lend themselves easily to manual construction and inspection. However, we see no *a priori* reason why our results would not extend to other architectures such as transformers or convolutional networks, given the generality of the MDL principle, and the fact that it has been shown to be beneficial across various domains and learning tasks, including linguistic phenomena (see Stolcke, 1994, Grünwald, 1996, de Marcken, 1996, and Rasin et al., 2021, among others).

6 Limitations

In Section 4.4 we explored the loss surface around the golden network using different objectives, and found that using L1/L2 regularization leads to sub-optimal networks lying at optimum points, while using the MDL objective leads to the golden network lying at an optimum point. However, since the loss exploration is not (and cannot be) exhaustive, caution is needed when making generalizations based on these results.

First, when using L1/L2, it is still possible of course that better optima lie somewhere else in the loss spaces, and that the respective minimizing networks have perfect performance. However, given the discussion in Section 3.1 about scalar magnitude vs. simplicity, we find this possibility unlikely, but admittedly still possible.

Conversely, when using the MDL objective, here it is conceivable that other networks would have better MDL scores and suboptimal performance. While this cannot be ruled out completely, we believe that using the MDL objective accompanied by a reasonable encoding scheme like the one used here makes over/under-fitting unlikely. This is arguably not the case for L1/L2. (Another possibility, that of a network with a better MDL score but equivalent perfect performance, is more likely, given that the golden network was manually designed and can potentially be optimized further.)

Finally, a major practical limitation of the current work relates to the non-differentiability of the MDL objective. This is especially problematic for ANNs, given that current standard training methods rely almost exclusively on gradient descent. One could then consider L1/L2 as a differentiable proxy for a strict formalization of simplicity. However, the current work sheds light on the shortcomings of these compromises. This in turn could lead both to a more informed use of such proxies, and potentially to further research regarding better optimization techniques for MDL.

7 Acknowledgements

This project was provided with computer and storage resources by GENCI at IDRIS thanks to the grant 2023-AD011013783R1 on the supercomputer Jean Zay’s V100 partition.

References

- Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. [On the Ability and Limitations of Transformers to Recognize Formal Languages](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.
- Gregory J. Chaitin. 1966. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569.
- Carl de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, MIT, Cambridge, MA.
- Nadine El-Naggar, Pranava Madhyastha, and Tillman Weyde. 2022. [Exploring the Long-Term Generalization of Counting Behavior in RNNs](#).
- Nadine El-Naggar, Andrew Ryzhikov, Laure Daviaud, Pranava Madhyastha, and Tillman Weyde. 2023. Formal and empirical studies of counting behaviour in ReLU RNNs. In *Proceedings of 16th Edition of the International Conference on Grammatical Inference*, volume 217 of *Proceedings of Machine Learning Research*, pages 199–222. PMLR.
- Jeffrey Elman, L. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.
- Felix Gers and Jürgen Schmidhuber. 2001. [LSTM recurrent networks learn simple context-free and context-sensitive languages](#). *IEEE Transactions on Neural Networks*, 12(6):1333–1340.
- Peter Grünwald. 1996. A minimum description length approach to grammar inference. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, Springer Lecture Notes in Artificial Intelligence, pages 203–216. Springer.
- Geoffrey E. Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13.
- Sepp Hochreiter and Jürgen Schmidhuber. 1994. Simplifying Neural Nets By Discovering Flat Minima. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Armand Joulin and Tomas Mikolov. 2015. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Diederik P. Kingma and Jimmy Ba. 2017. [Adam: A Method for Stochastic Optimization](#).

- Andrei Nikolaevic Kolmogorov. 1968. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2:157–168.
- Nur Lan, Emmanuel Chemla, and Roni Katzir. 2023. Benchmarking Neural Network Generalization for Grammar Induction. In *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*, pages 131–140, Gothenburg, Sweden. Association for Computational Linguistics.
- Nur Lan, Michal Geyer, Emmanuel Chemla, and Roni Katzir. 2022. **Minimum Description Length Recurrent Neural Networks**. *Transactions of the Association for Computational Linguistics*, 10:785–799.
- Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.
- Ming Li and Paul Vitányi. 2008. **Chapter 1.4, Binary Strings**. In *An Introduction to Kolmogorov Complexity and Its Applications*, Texts in Computer Science. Springer New York, New York, NY.
- William Merrill. 2021. **On the Linguistic Capacity of Real-Time Counter Automata**.
- William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. **A Formal Hierarchy of RNN Architectures**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 443–459, Online. Association for Computational Linguistics.
- Ezer Rasin, Iddo Berger, Nur Lan, Itamar Shefi, and Roni Katzir. 2021. **Approaching explanatory adequacy in phonology using Minimum Description Length**. *Journal of Language Modelling*, 9(1):17–66.
- J. Rissanen. 1978. **Modeling by shortest data description**. *Automatica*, 14(5):465–471.
- Jürgen Schmidhuber. 1997. **Discovering Neural Nets with Low Kolmogorov Complexity and High Generalization Capability**. *Neural Networks*, 10(5):857–873.
- Hava T. Siegelmann and Eduardo D. Sontag. 1992. **On the computational power of neural nets**. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT ’92*, pages 440–449, New York, NY, USA. Association for Computing Machinery.
- Ray J. Solomonoff. 1964. A formal theory of inductive inference, parts I and II. *Information and Control*, 7(1 & 2):1–22, 224–254.
- Andreas Stolcke. 1994. *Bayesian Learning of Probabilistic Language Models*. Ph.D. thesis, University of California at Berkeley, Berkeley, California.
- Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2023. **Transformers as Recognizers of Formal Languages: A Survey on Expressivity**. *arXiv preprint arXiv:2311.00208*.
- Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. **LSTM Networks Can Perform Dynamic Counting**. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.
- Whitney Tabor. 2000. Fractal encoding of context-free grammars in connectionist networks. *Expert Systems*, 17(1):41–56.
- Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the Practical Computational Power of Finite Precision RNNs for Language Recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745.
- Byoung-Tak Zhang and Heinz Muhlenbein. 1993. Evolving optimal neural networks using genetic algorithms with Occam’s razor. *Complex systems*, 7(3):199–220.

A Grid search hyper-params

Training size: 500/1000/5000/10000. Random seed: 100/200/300/400/500. Regularization: none/L1/L2. Regularization lambda when relevant: 0.1/0.5/1.0. Dropout rate: none/0.2/0.4/0.6. Early stop after no improvement for number of epochs: none/2/10. Weight initialization: uniform/normal.

All simulations used the Adam optimizer (Kingma and Ba, 2017) with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and ran for 20,000 epochs unless stopped by early stopping.

B Golden $a^n b^n$ LSTM construction

This section spells out the construction of the optimal $a^n b^n$ network from Section 4. The network is designed to output the correct probability distribution for $a^n b^n$ strings induced by the PCFG in (2). The target probabilities are plotted in Figure 2a.

The general idea is to implement a counting mechanism in the LSTM cell and then to pass this value through a linear layer and a softmax, which outputs the target probabilities. A full PyTorch implementation of the network is given at <https://github.com/0xnurl/mdl-lstm>.

B.1 Representations and constants

We use a standard LSTM cell represented by the following functions:

Loss	λ	Golden net		Best trained net		Best trained, no early stopping	
		Loss	Test acc. %	Loss	Test acc. %	Loss	Test acc. %
CE	-	3.58e-01	100.00	3.58e-01	77.33	3.57e-01	64.97
CE + L1	0.1	2.50e+02	100.00	1.67e+01	77.21	1.85e+01	86.73
	0.5	1.24e+03	96.23	8.17e+01	77.21	8.94e+01	96.24
	1.0	2.48e+03	0.00	1.63e+02	0.13	1.78e+02	96.24
CE + L2	0.1	3.72e+04	99.87	4.23e+01	77.21	5.80e+01	90.34
	0.5	1.86e+05	99.87	2.10e+02	70.38	2.85e+02	91.81
	1.0	3.72e+05	99.87	4.20e+02	70.38	5.69e+02	91.81
MDL	-	3.92e+03	100.00	5.88e+03	77.98	5.87e+03	69.68

Table 1: Minimum training loss values and best test accuracy scores in the space surrounding the following networks: the golden network, the best trained network which used early stopping, and the best trained network that was trained without early stopping or regularization terms. Bold indicates winning values for each row. Minimizing the loss and achieving perfect accuracy coincide only for the MDL objective.

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \quad (6)$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \quad (7)$$

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \quad (8)$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \quad (9)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (10)$$

$$h_t = o_t \odot \tanh(c_t) \quad (11)$$

where σ is the sigmoid activation and \odot is the element-wise product.

In the following construction, all weights are set to 0 unless mentioned otherwise.

The LSTM gates (sigmoids and tanh’s) need to be saturated in order to prevent leakage and keep the solution stable (Weiss et al., 2018). For this, a large enough input needs to be used. We select empirically:

$$LARGE = 2^7 - 1$$

which is the largest unsigned integer that fits in 7 bits (instead of, e.g., 2^7 , in order to save bits when using the encoding scheme from Section 3.2).

Network inputs and outputs are vectors of size 3, with the following class positions: $[\#, a, b]$, where $\#$ is the start/end-of-sequence symbol. Inputs are one-hot encoded, so that:

$$x_t = [\mathbb{1}_{\#}, \mathbb{1}_a, \mathbb{1}_b] \quad (12)$$

The notation $[\cdot \cdot \cdot]_{\tanh}$ is used as a shorthand for $\tanh([\cdot \cdot \cdot])$. Column vectors are printed as row vectors and omitting the transpose for readability.

B.2 Counting

The network’s memory vector c_t is of size 3. We describe the construction that leads to c_t holding the following target values at each time step:

$$c_t = [1, 1, \#a - \#b] \quad (13)$$

where $\#a$ and $\#b$ represent the number of a ’s and b ’s seen so far. $\#a - \#b$ thus counts the number of unmatched a ’s and becomes 0 only when the a ’s and b ’s are balanced. The two constant 1’s will be used downstream.

We first set:

$$W_{ig} = LARGE \cdot \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

Then, from definitions (8) and (12) and because all other weights feeding g_t are 0, we get:

$$g_t = \tanh(W_{ig}x_t) = [\mathbb{1}_{\#}, \mathbb{1}_{\#}, \mathbb{1}_a - \mathbb{1}_b] \quad (14)$$

i.e., the last component of g_t holds $+1$ when seeing ‘ a ’ or -1 when seeing ‘ b ’. 1’s are stored in the other components when the start-of-sequence symbol first appears.

Then, the input and forget gates are saturated to make the addition between g_t and c_{t-1} stable. Saturating the gates is done through their biases in order to save on encoding length:

$$b_{ii} = b_{if} = LARGE \cdot [1, 1, 1]$$

We write off the saturated gates from definition (10), and get the recurrent update of the memory vector:

$$c_t = c_{t-1} + g_t \quad (15)$$

It is then simple to apply the recurrence and get the correct counting targets (13) for all time steps: (14) gives $g_0 = [1, 1, 0]$ for the first time step and $g_{t>0} = [0, 0, \mathbb{1}_a - \mathbb{1}_b]$ for all other steps.

B.3 Hidden vector

The following construction leads to the hidden vector h_t holding the following target values, representing the different phases of an $a^n b^n$ string:

$$h_t = [\mathbb{1}_\#, \mathbb{1}_a, \mathbb{1}_{\#a>\#b}] \tanh \quad (16)$$

We first construct o_t as a mask vector to select the relevant part from c_t in (13) based on the current phase of the string.

The mask o_t is constructed by setting:

$$W_{io} = \text{LARGE} \cdot \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$$b_{io} = \text{LARGE} \cdot [-1, -1, -1]$$

Since all other weights in definition (9) are 0, we get:

$$o_t = \sigma(W_{io}x_t + b_{io})$$

Following the indexing of x_t , this results in a one-hot mask based on the current input:

$$o_t = [\mathbb{1}_\#, \mathbb{1}_a, \mathbb{1}_b]$$

(W_{io} and b_{io} are used instead of setting W_{io} to the identity because of the sigmoid in (9).)

Combined with (13) we get:

$$h_t = o_t \odot \tanh(c_t) = [\mathbb{1}_\#, \mathbb{1}_a, \mathbb{1}_{\#a>\#b}] \tanh \quad (17)$$

This vector is (tanh-)one-hot in all cases except when $\#_a = \#_b$, in which case it zeros-out.

B.4 Output layer

The hidden vector h_t is then multiplied by a linear layer W_{out} . We build the values of W_{out} backwards based on the optimal target probabilities.

Each $a^n b^n$ string has four phases: the start-of-sequence step, the ‘a’ phase, the $b^{m<n}$ phase, and the final-‘b’ phase. Each row in the following matrix holds the optimal probabilities for the respective phase, based on PCFG (2):

$$\text{Targets} = \begin{pmatrix} p & 1-p & 0 \\ 0 & 1-p & p \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (18)$$

Since the output layer feeds a final softmax, we build the logits backwards:

$$W_{log} = \ln(\text{Targets} + \varepsilon)$$

with ε preventing taking the log of 0. Here we use $\varepsilon = (2^{14} - 1)^{-1}$.

Since we have four states and only three components in the hidden vector, we superimpose the four states onto three. First, split W_{log} into $W_{out'}$ which contains the first three states, and a bias b_{out} which contains the fourth:

$$W_{out'} = W_{log_{1:3}}$$

$$b_{out} = W_{log_4}$$

Then subtract to get:

$$W_{out''} = (W_{out'} - b_{out})^T$$

The transpose is taken so that multiplying by the one-hot h_t copies columns.

Finally, divide by $\tanh(1)$ because h_t is tanh-one-hot based on (17):

$$W_{out} = W_{out''} / \tanh(1)$$

As seen in (17), h_t is one-hot during all phases except the last ‘b’, and thus copies the relevant probability distribution from (18). Adding the bias undoes the superimposition. h_t is all-zero only when $\#_a = \#_b$, in which case $W_{out} \cdot h_t$ is zero. In this case the probabilities for the fourth state (final-b), stored in b_{out} , are outputted.

Chapter 4

Large Language Models and the Argument From the Poverty of the Stimulus (joint with Emmanuel Chemla and Roni Katzir)

Nur Lan, Emmanuel Chemla, and Roni Katzir. 2024. Large Language Models and the Argument From the Poverty of the Stimulus. To appear in *Linguistic Inquiry*.

Large Language Models and the Argument From the Poverty of the Stimulus*

Nur Lan, Emmanuel Chemla, and Roni Katzir

March 2024

Abstract

How much of our linguistic knowledge is innate? According to much of theoretical linguistics, a fair amount. One of the best-known (and most contested) kinds of evidence for a large innate endowment is the so-called *argument from the poverty of the stimulus* (APS). In a nutshell, an APS obtains when human learners systematically make inductive leaps that are not warranted by the linguistic evidence. A weakness of the APS has been that it is very hard to assess what is warranted by the linguistic evidence. Current Artificial Neural Networks appear to offer a handle on this challenge, and a growing literature over the past few years has started to explore the potential implications of such models to questions of innateness. We focus here on Wilcox et al. (2023), who use several different networks to examine the available evidence as it pertains to *wh*-movement, including island constraints. They conclude that the (presumably linguistically-neutral) networks acquire an adequate knowledge of *wh*-movement, thus undermining an APS in this domain. We examine the evidence further, looking in particular at parasitic gaps and across-the-board movement, and argue that current networks do not, in fact, succeed in acquiring or even adequately approximating *wh*-movement from training corpora that roughly correspond in size to the linguistic input that children receive. We also show that the performance of one of the models improves considerably when the training data

*Acknowledgments: We thank Bob Berwick, Noam Chomsky, Milica Denić, Danny Fox, Colin Phillips, Ezer Rasin, Dominique Sportiche, Imry Ziv, and an anonymous reviewer for *LI*, as well as the audience at Tel Aviv University. This project was provided with computer and storage resources by GENCI at IDRIS thanks to the grant 2023-AD011013783R1 on the supercomputer Jean Zay's V100 partition.

are artificially enriched with instances of parasitic gaps and across-the-board movement. This finding suggests, albeit tentatively, that the failure of the networks when trained on natural, unenriched corpora is due to the insufficient richness of the linguistic input, thus supporting the APS.

1 Background: Innateness and the Argument From the Poverty of the Stimulus

One way in which linguists have argued that humans are born with nontrivial biases is through cases where speakers' linguistic knowledge goes beyond what seems warranted by the data they were exposed to. If humans systematically arrive at this knowledge given the data while linguistically-neutral learners exposed to similar data do not, then humans are not linguistically neutral: they come to the task of language acquisition prepared. Reasoning of this kind is known as an *argument from the poverty of the stimulus* (APS), and since its introduction by Noam Chomsky over 50 years ago it has been central to the study of the human linguistic capacity.^{1,2} Here we will focus on one APS, concerning wh-movement, but various other APSs have been discussed in the literature based on a range of empirical phenomena such as *one*-substitution (introduced in Baker 1978), subject-auxiliary inversion (introduced in Chomsky 1971), and plurals within compounds (introduced in Gordon 1985).

The APSs just mentioned (and others like them) have been taken to argue for nontrivial innate biases in humans. For example, the APS from subject-auxiliary inversion has been taken to support an innate bias for hierarchical transforma-

¹The general considerations behind the APS are discussed already in chapter 1 of Chomsky 1965. Further considerations are discussed in Chomsky 1971, pp. 26–8, Chomsky 1975, pp. 30ff., and Chomsky, 1980, pp. 42ff., as well as in much subsequent work.

In addition to the APS, linguists have also identified other sources of evidence supporting the innateness of nontrivial linguistic knowledge. For example, there are arguments from the *richness* of the stimulus, where a pattern that is clearly represented in the input data and would be easily picked up by a linguistically-neutral learner is simply ignored by human learners. Evidence from typological asymmetries has also played a very important role in linguistic reasoning. A proper discussion of such sources of evidence falls outside the scope of the present paper, and in what follows we focus exclusively on the APS.

²Throughout the discussion we set aside the question of whether the knowledge under consideration is specific to linguistics (and, if so, how much of it is purely syntactic) or whether it is shared with other cognitive domains. Our sole focus is on whether a neutral learner would be justified in acquiring the relevant knowledge based on a given linguistic input.

tions over linear ones. The APS from wh-movement that we discuss below will similarly support an intricate bias that a linguistically-neutral learner is not expected to have. The same holds for other APSs in the literature. In this, these APSs go beyond the early observation that children can produce and understand unboundedly many sentences after encountering only a finite number of sentences (Chomsky 1957, p. 15). While generalizing from a finite input to an infinite language is perhaps not entirely trivial, it is something that most learning algorithms do. And importantly, this ability does not imply any biases that a linguistically-neutral learner will not have.

While the APS has been central to linguistic reasoning, it has also generated much controversy. Contesting a given APS requires challenging either the knowledge attained by humans or the information available to the child learner. It is the latter that often comes under attack. The reason for this vulnerability is that it is extremely difficult to assess what information exactly is available to the child over the relevant time period (often years of exposure) and hard to tell what a general-purpose, linguistically-neutral learner would do with this kind of information. One can try to look for pieces of evidence that seem relevant for the knowledge at stake — e.g., as done for the case of subject-auxiliary inversion in English by Legate and Yang (2002) — but as noted by Lewis and Elman (2001), Perfors et al. (2011), and others, this methodology runs the risk of underestimating the available information: even if we fail to find the evidence we are looking for, a general-purpose learner might be able to take advantage of other sources of information. This methodology also risks *overestimating* the available information: even if we find several instances of the evidence we are after, a general-purpose learner might treat those instances as noise and fail to draw the inference that we intuitively expect it to. In the absence of an actual learner that can use the information that is available in an entire corpus it is just very hard to estimate whether the data support the knowledge under consideration.³

How then can we reason about the information available to the child and ask

³See Pullum and Scholz (2002), Lidz et al. (2003), Foraker et al. (2009), Hsu and Chater (2010), Berwick et al. (2011), Perfors et al. (2011), and Pearl and Sprouse (2013), among others, for much relevant discussion.

In studies of analogous inductive leaps in other species, this worry regarding the input has been addressed by controlling the information available to the learners (see, e.g., Dyer and Dickinson 1994). To a certain extent this can be done with humans in experiments of artificial-grammar learning (see, e.g., Wilson 2006). But for the main APSs in the literature, which concern the normal course of child language acquisition, controlling the information available to the learner is not an option.

whether it suffices to support the acquisition of a given piece of knowledge by a linguistically-neutral learner? In an ideal world, one would (a) take a sufficiently powerful learner that can be seen to not be biased in favor of the relevant knowledge; (b) train this learner on a corpus that corresponds to the linguistic input that children receive; and (c) check whether the learner has indeed acquired the knowledge under consideration. In such an ideal world, one might perhaps be able to work with an induction algorithm for unrestricted (type-0) grammars, or for a general-purpose programming language such as Python (e.g., focusing on *acceptors*, programs that accept some strings over a given alphabet and reject or enter an infinite loop on the rest). These (equally powerful) formalisms are capable of representing the kinds of knowledge that linguists consider but can be seen as linguistically neutral. In our case, while both unrestricted grammars and Python programs can easily represent the equivalent of wh-movement, including the intricacies of islands, nothing about either framework seems to favor *a priori* such representations. One can of course consider other representational frameworks, including less powerful ones (e.g., context-sensitive grammars) as long as they can still represent linguistic knowledge but are not biased in its favor. One would still need to ensure that the learning algorithm itself does not bias the learner for or against linguistic patterns, but this can be done in various ways, such as by using a linguistically-neutral prior within a Bayesian learner. After training on a developmentally-realistic corpus, corresponding to a few years of human linguistic experience, the knowledge acquired by the algorithm can then be directly inspected at stage (c).

In the actual world, combining (a) through (c) is currently impossible. For many years, the combination of (a) and (b) was already a major barrier. General program induction algorithms of the kind just mentioned, for example, address (a) but fail on (b), since they are limited to very small training corpora. On the other end of the scale, *n*-gram models can easily be trained on very large corpora, thus addressing (b), but their representational capacity is much too limited to capture or even to adequately approximate linguistic knowledge such as wh-movement. Other models, such as probabilistic context-free grammars, fall between these two extremes but still typically struggle with the combination of (a) and (b) when it comes to patterns such as wh-movement.

The challenge of assessing the information available to the child has become less of an obstacle lately, with the advent of Large Language Models (LLMs). These models, which rely on modern architectures of artificial neural networks (ANNs), do not yet fully address any of (a) through (c) — a matter that has been discussed in recent literature and that we return to below — but they can be trained

on very large corpora and are generally quite successful in acquiring sequential dependencies.⁴ This has allowed a large and growing literature to use these models to ask questions relating to the learning of linguistic knowledge by LLMs, often with specific reference to the APS. Of particular relevance to our purposes here is work starting with Linzen et al. (2016) and including Bernardy and Lapin (2017), Chowdhury and Zamparelli (2018), Gulordava et al. (2018), Kuncoro et al. (2018), Marvin and Linzen (2018), Wilcox et al. (2018, 2019, 2023), Bhattacharya and van Schijndel (2020), Chaves (2020), Warstadt et al. (2020), Huebner et al. (2021), Ozaki et al. (2022), and Yedetore et al. (2023), among others, that examines the preference of LLMs within minimal pairs. Here we focus on the application of LLMs to the domain of wh-movement, following Wilcox et al. (2018, 2019, 2023), Chowdhury and Zamparelli (2018), Bhattacharya and van Schijndel (2020), Chaves (2020), Warstadt et al. (2020), and Ozaki et al. (2022). In particular, we examine the claim by Wilcox et al. (2023; WFL) that current models debunk an APS in this domain: one that says that the input is insufficiently rich to allow a general-purpose learner to acquire wh-movement.⁵

The present paper extends WFL’s probing of LLMs’ knowledge of wh-movement, arriving at conclusions that are at odds with those of WFL. We start, in Section 2, with a brief overview of the general setup for the rest of the paper. Among other things we discuss how LLMs can be used as tools for assessing the information in a given corpus without assuming that these models are cognitively plausible in any way and without even asking whether these models have achieved an adequate knowledge of the pattern under consideration.⁶ Rather, we treat these models as

⁴Long before the current models, earlier ANN architectures were used in debates of the APS, and in particular in attempts to argue against various versions of it (see Elman et al. 1996, Lewis and Elman 2001, and Reali and Christiansen 2005, among others, and see Berwick et al. 2011 for a critical analysis of some earlier attempts). Early ANNs, however, were limited in their capacities and generally trained on small corpora, and it is unclear whether they could be used to reason about whether a corpus that roughly corresponds to children’s linguistic exposure supports the acquisition of complex grammatical knowledge. In this sense, these earlier models were not yet capable of addressing the combination of (a) and (b). The ability of current models to train on realistically large corpora is a helpful step towards using them constructively in debates about the APS.

⁵See Pearl and Sprouse (2013) and Phillips (2013) for earlier discussion of APS in the context of acquiring islands.

⁶Our results do bear on the question of the cognitive plausibility of LLMs, however. In particular, since our results are negative they provide further evidence, if such was needed, that current LLMs are not cognitively plausible models of human linguistic cognition, *contra* Piantadosi (2023). See Katzir (2023), Kodner et al. (2023), Moro et al. (2023), and Rawski and Baumont (2023), among others, for additional discussion.

proxies for future learners and ask only whether these proxies provide a reasonable approximation of the target pattern. In Section 3 we discuss the success of LLMs in simple cases of wh-dependencies, as noted by WFL. In Section 4 we show that the scope of the LLMs’ success is rather limited. In particular, LLMs fail to adequately approximate human knowledge of a much-studied family of cases, falling under the labels of parasitic gaps and across-the-board movement, in which certain additional gaps make an otherwise problematic gap inside an island acceptable. It is cases such as these that are typically taken by linguists to suggest an APS, and our findings show that the performance of current LLMs does not, in fact, debunk this APS. In Section 5, we ask whether the LLMs fail only due to their own limitations or whether their failure reflects also the insufficient richness of their training data. We address this question by retraining one of the models on corpora that are clearly *not* impoverished with respect to the relevant patterns and showing that the performance of the model improves significantly on the enriched corpora. This, in turn, strengthens the APS, if also tentatively. Section 6 concludes.

2 The General Setup

Simplifying considerably, a *gap*, such as the missing complement of *with* in (1a) and (1c), appears if and only if it is preceded by an appropriate *filler*, such as the wh-phrase *who* in (1a) and (1b). When there is both a filler and a gap (1a) or neither (1d) the result is good; when there is a filler and no gap (1b) or a gap and no filler (1c) the result is bad.⁷

- (1) a. I know who you talked with __ yesterday. (+*filler*,+*gap*)
 b. * I know who you talked with Mary yesterday. (+*filler*,−*gap*)
 c. * I know that you talked with __ yesterday. (−*filler*,+*gap*)
 d. I know that you talked with Mary yesterday. (−*filler*,−*gap*)

There is much further nuance to wh-movement, some of which we will briefly mention below. For now, let us consider how one might check if the input data are rich enough for a linguistically-neutral learner to acquire the knowledge of wh-movement. We mentioned earlier that in an ideal world, we could try to evaluate a given APS by (a) taking a sufficiently powerful learner that can be seen to not

⁷In order to make it possible to alternate the \pm *filler* condition, and following WFL, we embed the relevant examples under *I know*: *I know who...* (+*filler*) vs. *I know that...* (−*filler*).

be biased in favor of the relevant knowledge; (b) training it on a developmentally-realistic corpus; and (c) checking whether the learner has indeed acquired the knowledge under consideration. We also mentioned that current LLMs do not quite handle any of (a)–(c). In the remainder of this section we will review some of the shortcomings of LLMs with respect to each of (a)–(c) and discuss how LLMs can still be helpful (if also inconclusive) in studying the APS.

2.1 Powerful and Unbiased?

We do not know how powerful LLMs are. Representationally, recurrent neural networks were shown to be Turing-complete under idealized assumptions of infinite precision and computation time (Siegelmann and Sontag, 1991, 1995). Under realistic assumptions, however, the representational capacity of recurrent neural networks is much more limited, as shown for example by Weiss et al. (2018) and Merrill et al. (2020). A similar situation obtains with the more recent Transformer architecture (see survey in Strobl et al., 2023). Moreover, even this limited representational capacity of ANNs under realistic assumptions is often not attained in practice, and there is evidence suggesting that standard training methods prevent at least some models from acquiring key patterns (see El-Naggar et al. 2023 and Lan et al. 2022, 2023, 2024).

Given these limitations we will avoid assuming that current models can learn the pattern of wh-movement and only rely on their ability to provide a reasonable approximation of the pattern. If a given ANN can reach such an approximation from a sufficiently rich corpus, we can use it as a proxy for a good general-purpose learner, even if the ANN is not such a learner itself. We can then use the ANN to study the APS. If the model provides a reasonable approximation of wh-movement from a developmentally-realistic corpus, this suggests that a good general-purpose learner could learn the correct pattern from that corpus and that the APS in this domain does not hold. And if the model fails to reach such an approximation this suggests that a good general-purpose learner might not learn the correct pattern from that corpus and that the APS in this domain stands.

The use of ANNs as proxies still requires understanding how their biases relate to the approximations of the relevant linguistic patterns. Unfortunately, due to how poorly these models are understood, we cannot say with any certainty whether a given ANN is linguistically-neutral, and if not, whether its biases push it in the direction of a given linguistic pattern. Until more is known about these biases, and as correctly cautioned by Rawski and Heinz (2019), any claims about the neutrality of these models must be taken as tentative. Still, it strikes us as reason-

able to assume that current LLMs are not particularly biased *against* the linguistic dependencies under consideration. This is especially so since these models have been developed over the past decades so as to succeed in capturing key patterns in linguistic sequences; therefore, if they do have linguistically-relevant biases after all, those are likelier to be in favor of the patterns under consideration rather than against them. Consequently, if the models fail to acquire an adequate approximation of the relevant dependencies, this failure can be taken to be informative. More directly, and as mentioned above, we will show in Section 5 that with richer training data, at least one model improves its approximation of the pattern of wh-movement, which will suggest that the failure of the model on its original training data is not due solely to its biases and other limitations but also to the lack of sufficient evidence in the data.

2.2 Training on Developmentally-Realistic Corpora?

As discussed in detail by Warstadt and Bowman (2022), current models are not trained on developmentally-realistic corpora. Such a corpus would be the equivalent of the relevant input that a child receives over the first few years of their life. But the training data for current models are more informative than the input to the child in some ways and less informative in others. They are more informative, for example, in that they are orders of magnitude larger than what humans are exposed to in a whole lifetime. They are less informative in that they are purely textual and do not reflect environmental and social cues, prosody, and input from modalities other than speech, all of which are in principle available to children. See Warstadt and Bowman (2022) for further discussion.

The particular pattern that we discuss here can arguably be investigated on the basis of the information available in standard training corpora. Of course, this is not to say that the dependencies under consideration do not depend on extra-textual cues (a matter of ongoing discussion in the literature). But if, as WFL suggest and as our results further support, the basic pattern of wh-movement can be approximated based on text, there is no reason to think that the further approximation of parasitic gaps and across-the-board movement will crucially require extra-textual cues. This point will be reinforced by the evidence from retraining in Section 5.

As to the size and quality of the text in our training data, we use a range of corpora, reviewed immediately below, that span the spectrum from the very small (CHILDES) through mid-size (Wikipedia) to the very large (the training sets for GPT-2/3/j). We do so in an attempt to make up for the inadequacy of individual

corpora to some extent, but we acknowledge that this is at best a partial remedy.

The models we use in our evaluation are the following, also summarized in Table 1: an LSTM and a Transformer from Yedetore et al. (2023), both of which were trained on the CHILDES corpus of child-directed speech (MacWhinney, 2014);⁸ an LSTM trained on English Wikipedia (Gulordava et al., 2018); a Transformer trained on English Wikipedia;⁹ Open AI’s GPT-2 (Radford et al., 2019); GPT-j (Wang and Komatsuzaki, 2021); and OpenAI’s GPT-3 (Brown et al., 2020).¹⁰ The LSTM trained on English Wikipedia and both GPT-2 and GPT-3 are used by WFL in their evaluation.¹¹

In order to get a very rough sense of the number of years of linguistic experience that a given training corpus corresponds to we follow common practice (used also by WFL) based on Hart and Risley (1995)’s estimates about the number of words that American children typically hear during acquisition. According to these estimates, the models just mentioned were exposed to amounts of data ranging from ten months of linguistic experience (CHILDES LSTM and Transformer)

⁸The models in Yedetore et al. (2023) were trained on utterances of 52 children between the ages of six months to 12 years, from the North American English subset of the CHILDES corpus. The total training size amounts to 9.6 million words, which is considerably less than what children typically receive by the time they exhibit knowledge of the pattern under consideration here. Qualitatively, on the other hand, this training corpus is arguably more realistic than the much larger training corpora used for the remaining models.

Out of ten models per architecture (LSTM/Transformer) trained in Yedetore et al. (2023) with different random seeds, we use the model with the best test perplexity.

⁹We added this Transformer since we wanted to evaluate the information in the English Wikipedia training corpus (the most realistic developmentally in terms of size of all the training corpora under consideration) using a more current architecture than the LSTM that WFL use. We used one of the large Transformer architectures used in Yedetore et al., 2023: 8 layers, hidden and embedding size 1600, and 16 attention heads, trained using the same training regime. Since the current task is limited to single sentences, we lowered the Transformer’s context size to 30 (compared to 500 in Yedetore et al. 2023), closer to the average sentence size in the Wikipedia dataset (27.2).

¹⁰Model version *text-davinci-003*, the latest supported version not fine-tuned using reinforcement learning from human feedback (RLHF) for chat and other applications; however, the model is still trained with supervised fine-tuning, and it is proprietary. See <https://archive.today/2023.10.07-060351/https://platform.openai.com/docs/models/gpt-3-5> for OpenAI’s documentation retrieved October 2023 (archived snapshot).

¹¹WFL also use another LSTM, from Jozefowicz et al. (2016). We chose not to include that model in our evaluation since it is extremely slow to work with. For WFL’s evaluation, which used a small number of sentences, this was not a problem, but our evaluation relied on a much larger number of sentences, making Jozefowicz et al. (2016)’s model impracticable.

Model	~Tokens in training data	~Human equivalent
CHILDES LSTM (Yedetore et al., 2023)	8.6 million	10 months
CHILDES Transformer (Yedetore et al., 2023)		
Wikipedia LSTM (Gulordava et al., 2018)	90 million	8 years
Wikipedia Transformer		
GPT-2 (Radford et al., 2019)	8 billion	730 years
GPT-3 (Brown et al., 2020)	114 billion	10,300 years
GPT-j (Wang and Komatsuzaki, 2021)	402 billion	36,540 years

Table 1: Training data size of the seven language models considered here, and the human linguistic experience equivalent to these data sizes; human equivalents follow WFL (based on Hart and Risley 1995) who assume a daily exposure to ~30,000 words by children, or around 11 million words per year.

through eight years of linguistic experience (Wikipedia LSTM and Transformer) to between 10 and 500 human lifetimes (GPT-2, GPT-3, and GPT-j); see Table 1. WFL note that the linguistic experience of some of the models is well above that of children in terms of size and could thus weaken their argument against the APS in case of successful learning by the models. However, in the case of a negative result, as in the current work, a large training corpus only makes failures to learn more informative.

2.3 Inspecting LLM Knowledge?

As mentioned, LLMs are very opaque. While symbolic models such as context-free grammars (whether probabilistic or not) can generally be inspected directly so as to reason about the knowledge that they incorporate, inspecting LLMs in a similar fashion is not currently possible. One might try to follow standard practice in linguistics and study the knowledge of LLMs from the outside, by examining which sentences they accept. We could then check, for example, whether a particular LLM believes that a given continuation such as *yesterday* or *Mary* is

grammatical following a given prefix such as *I know who/that you talked with* in (1) above. Unfortunately, however, we cannot currently check whether an LLM takes a given sentence to be grammatical. In fact, it is not clear whether current models even have a notion of grammaticality to begin with.

What LLMs do tell us is how *likely* they consider any given continuation. The problem is that grammaticality and probability are generally very different notions. And while the two are correlated — many ungrammatical continuations are also unlikely on any sensible notion of probability, and grammatical continuations are sometimes probable — this correlation is far from perfect (see Chomsky 1957, Berwick 2018, and Sprouse et al. 2018, among others, for relevant discussion). In particular, many grammatical continuations are highly unlikely; for example, *splat* is a grammatical but unlikely continuation of *John would like to eat a freshly-made*. And in some cases an ungrammatical continuation can be likely; for example, *is* is a likely but ungrammatical continuation of *The keys to the cabinet*, an instance of so-called *agreement attraction* (see, e.g., Bock and Miller 1991 and Wagers et al. 2009).¹²

In the cases we are interested in here, however, probability and grammaticality are often quite well aligned, and — as in many other cases discussed in the literature mentioned earlier on evaluating LLMs on minimal pairs — it is easy to find examples such as (1) in which the grammatical continuation is significantly more probable than the ungrammatical one on any sensible notion of probability. So if we focus on such cases where grammaticality and probability are aligned, and if ANNs are sufficiently good learning models — at least, good enough to provide a crude approximation of the pattern under consideration — then we can use the probabilistic predictions of the resulting LLMs to evaluate the APS by comparing their probability assignments within minimal pairs. If a given LLM systematically assigns a much higher probability to the grammatical continuation, one potential explanation for this success is that the pattern of wh-movement is represented sufficiently well in the model’s training data for the model to approximate it. While it remains unclear, as mentioned above, whether current ANNs themselves have a representation of grammaticality as distinct from probability or whether they can learn the true pattern, their success when trained on developmentally-realistic cor-

¹²Agreement attraction is a performance error. Speakers make such errors when distracted or in a hurry but less so when given more time. ANNs do not make this distinction: when they give a higher probability to an ungrammatical continuation their response reflects a faulty knowledge rather than a resource problem. This serves to further illustrate the inadequacy of ANNs as models of linguistic cognition but does not pose a problem for our use of these models as a tool for assessing the informativeness of the input data.

pora would suggest that a good linguistically-neutral learner that does have such representational abilities might acquire the pattern. Conversely, if the LLM does not systematically assign a much higher probability to the grammatical continuation, one possible explanation for this failure is that the pattern of *wh*-movement is not sufficiently well represented in the input data to merit its approximation by the model. This, in turn, would suggest that a good linguistically-neutral learner will not acquire the pattern from the input data. In this way, LLMs — even if their representational inadequacies prevent them from providing more than a crude approximation of the pattern under consideration — can serve as useful proxies for future general-purpose learners and help us reason about the information available in the input data.

Care is needed in interpreting the performance of the models, even when treated as proxies for future learners. As Kodner and Gupta (2020) and Vázquez Martínez et al. (2023) note, clearly inadequate models can still pass current benchmarks of minimal pairs. More generally, it is possible for a model to either succeed by accident or fail by accident. As we discuss below, and in line with recent literature, we will try to lessen the worry of uninformative success or failure: in addition to using a wide range of models trained on many different corpora, as already mentioned above, we will also vary the lexical choices within our minimal pairs and also control to some extent for very local preferences that the models might have and that could obscure their approximation of the pattern of *wh*-movement. But these remain partial remedies, and any positive conclusions from the evaluation must be qualified accordingly. This worry does not affect our argument against WFL’s conclusions: in this case, WFL make the positive claim that LLMs refute the APS, and we show (in section 4) that current LLMs provide no basis for such a conclusion. But the worry does affect our attempt (in section 5) to show that LLMs strengthen the APS. While we try to make the case that the failure of at least one of the models reflects the poverty of the stimulus, our conclusions in this part must remain tentative.

3 LLMs Succeed in Very Simple Cases of *wh*-movement

How rich is the input, then, when it comes to filler-gap dependencies of the *wh* kind? In very simple cases such as (1) above, the LLMs considered by WFL assign a higher probability to the grammatical continuation than to the ungrammatical one. Above we mentioned that success in cases such as those considered here, where probability and acceptability are aligned, should involve not just a higher

probability to the grammatical continuation but a *much* higher one. However, in order to give the models a better chance of refuting the APS, we will adopt a very lenient criterion for success and only ask if the probability assigned to the grammatical continuation is higher than that assigned to the ungrammatical one, without taking into account how much higher it is. This will allow a network to be considered successful even if it prefers the grammatical continuation by the slightest of margins. This lenient condition for success will strengthen our conclusions from cases of failure, which we get to in the sections below: if a network fails even with this lenient condition of success, this failure can be taken seriously.

Here and below we will follow WFL (and the psycholinguistic literature that they build on) and illustrate using *surprisal* values, where the surprisal of x is $-\log P(x)$, which is simply the negative of the logarithmically-scaled probability of x .¹³ The lower the probability the higher the surprisal; when the probability approaches 0 the surprisal tends to infinity, and as the probability approaches 1 the surprisal tends to 0. Since higher probability corresponds to lower surprisal, support for the model will come from its assigning lower surprisal to a grammatical continuation than to an ungrammatical one, which, as mentioned, is what WFL indeed find in simple cases.

Figure 1 illustrates the preference of the models considered here for the grammatical continuation over the ungrammatical one in a very simple case by plotting surprisal values for sentences (1a) and (1b). All models assign a lower surprisal value (i.e., a higher probability) to the grammatical continuation *yesterday* in the gapped sentence than to *Mary*. This suggests (albeit weakly) that the input is sufficiently rich for a general-purpose learner to acquire from it an approximation of some basic aspects of wh-movement.

WFL further suggest that the LLMs go beyond the basic knowledge that fillers and gaps go hand in hand. Specifically, they claim that LLMs are aware of *islands* (Ross, 1967): configurations in which a gap is bad even if there is a filler upstream. We illustrate this with the following:

- (2) * I know who [[the question whether ___ jumped] surprised Mary yesterday].

¹³WFL's methodology includes looking not just at *+filler* cases, as in (1a) and (1b), but also at the corresponding *-filler* ones, as in (1c) and (1d). We will follow WFL in this in our discussion in sections 4.3 and 5 below, but for the present we will attempt to keep the presentation simple by considering only *+filler* pairs.

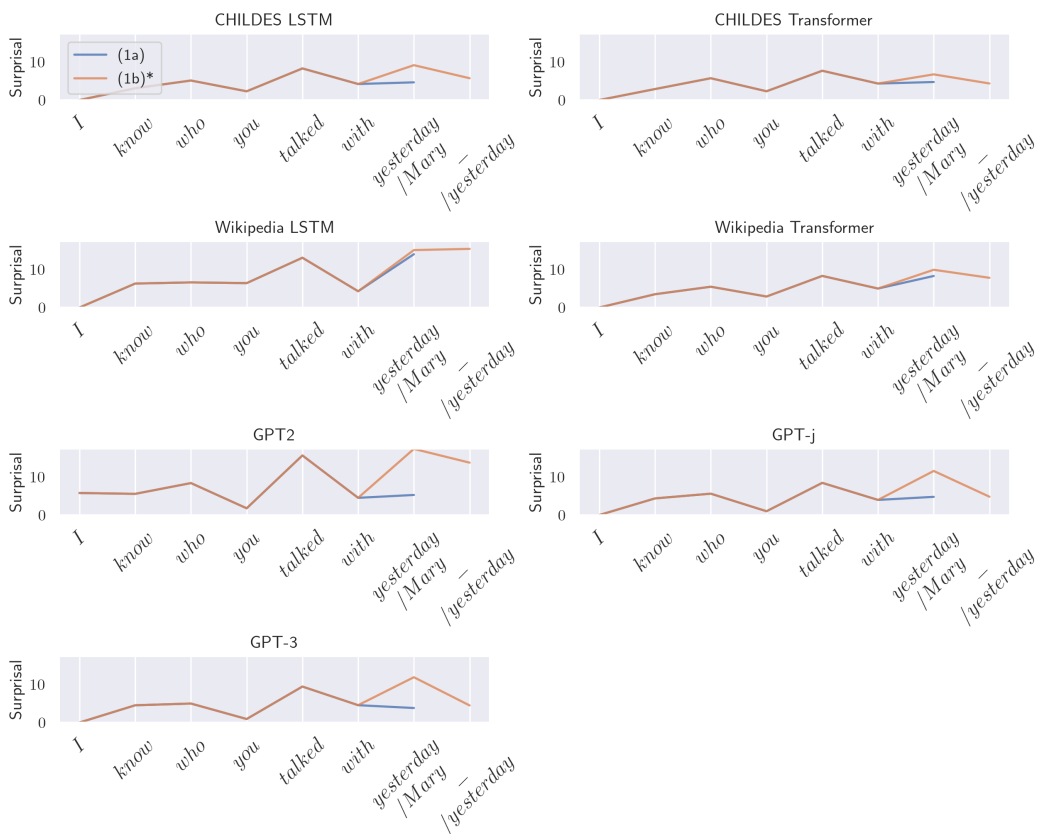


Figure 1: Raw surprisal values outputted by the LLMs for the grammatical (1a), in blue, and ungrammatical (1b), in orange. All models correctly output lower surprisal values for the grammatical continuation.

While, as discussed above, a filler upstream generally increases the LLMs’ expectation of a gap downstream, this expectation should be reduced within the subject of the embedded clause in (2). This subject is an island to movement, and extraction from within it is unacceptable and presumably highly unlikely. Figure 2 shows that the models are indeed surprised by the gap in (2), suggesting that their training corpora are informative with respect to this aspect of wh-movement.¹⁴

WFL consider a range of similar cases and conclude that linguistically-neutral

¹⁴The literature discusses various cases in which extraction from subjects (and other islands) is judged acceptable by speakers. Here and below we focus on relatively simple examples in which speaker judgments are clear, and our evaluation will concern the extent to which LLM preferences approximate these clear speaker judgments.

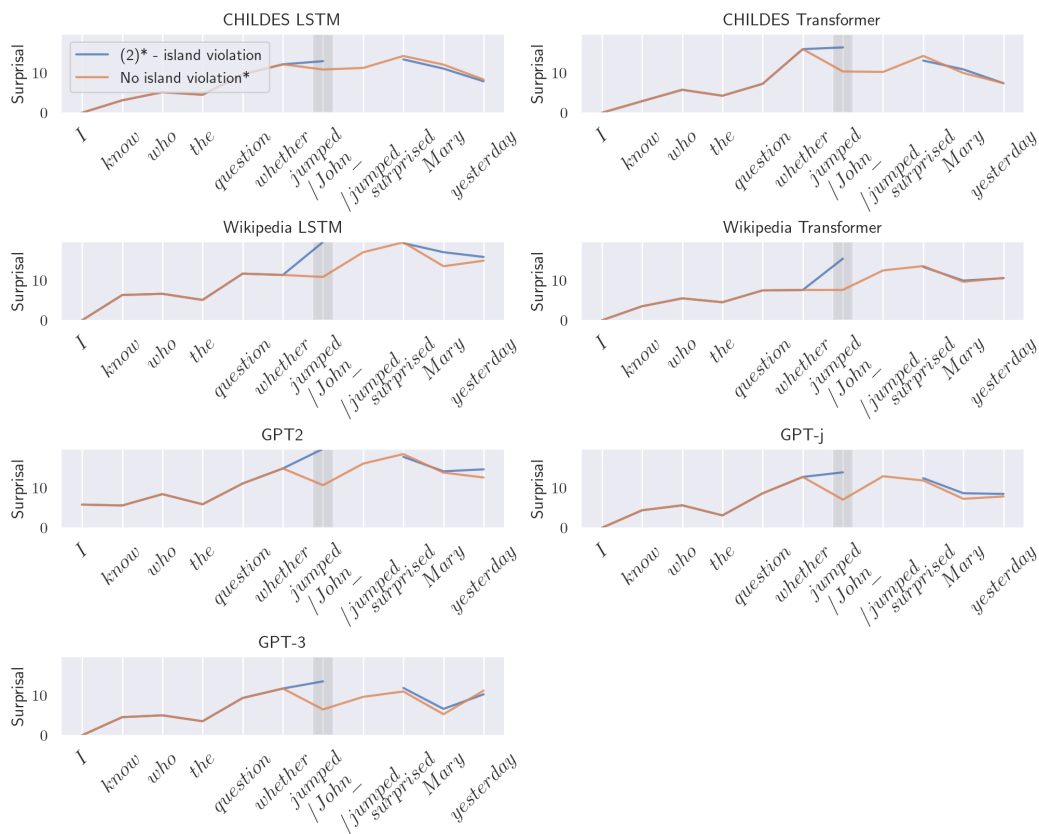


Figure 2: Raw surprisal values for the island violation sentence in (2), in blue, and a variant of the sentence with no island violation (we use *John* instead of the island-internal gap), in orange. All models are correctly surprised to find a gap within the island. Note that since the variant with *John* has no downstream gap that would correspond to the upstream filler, it is ungrammatical. For a grammatical version one could replace *Mary* with a gap. This matter, however, is orthogonal to the surprisal at the island-internal gap, which is what this figure illustrates.

learners can acquire the intricacies of wh-movement from the input data and that consequently the APS in this domain falls apart.

4 LLMs Fail on Slightly More Complex (But Still Simple) Cases of wh-movement

We now turn to a well-studied nuance of islands: in various cases, an otherwise impossible gap inside an island is made possible by a separate gap elsewhere. For example, while (3a), with a subject-internal gap, is bad, its counterpart in (3b), which has an added gap in the direct-object position of the main clause, is good. This phenomenon is known as a *parasitic gap* (PG): the gap inside the subject island becomes acceptable parasitically, based on the direct-object gap.¹⁵

- (3) a. *I know who [John's talking to __] is going to annoy you soon.
b. I know who [John's talking to __] is going to annoy __ soon.

Somewhat similarly, while (4a), with a gap inside a conjunct, is bad, its counterpart in (4b), where there is a gap in the other conjunct as well, is good. This phenomenon is known as *across-the-board movement* (ATB).¹⁶

- (4) a. *I know who John [met __ recently] and [is going to annoy you soon].
b. I know who John [met __ recently] and [is going to annoy __ soon].

4.1 An Initial Failure

Do LLMs approximate the patterns of PG and ATB? Both Wilcox et al. (2018) and Chaves (2020) mention PG and ATB in passing, but we are not familiar with attempts in the literature to evaluate the success of LLMs in approximating these patterns. Figures 3-4 illustrate that all the LLMs that we are considering here fail on (3a) and (4a), even on our very lenient condition of success: they do not just fail to assign a much higher probability to the grammatical continuation over the ungrammatical one in this simple case; they actually prefer the *ungrammatical* continuation. This seems to indicate that the ANNs have failed to acquire a good approximation of the relevant constructions, which in turn challenges WFL's

¹⁵Not all impossible gaps can be rescued in this way. For example, adding further gaps does little to improve (2) above.

¹⁶We set aside the important question of what stands behind PGs and ATB and whether the two are related. See Ross (1967), Williams (1977, 1990), Engdahl (1983), Haik (1985), Munn (1992), Postal (1993), Fox (2000), Nissenbaum (2000), and Hornstein and Nunes (2002), among others, for discussion.

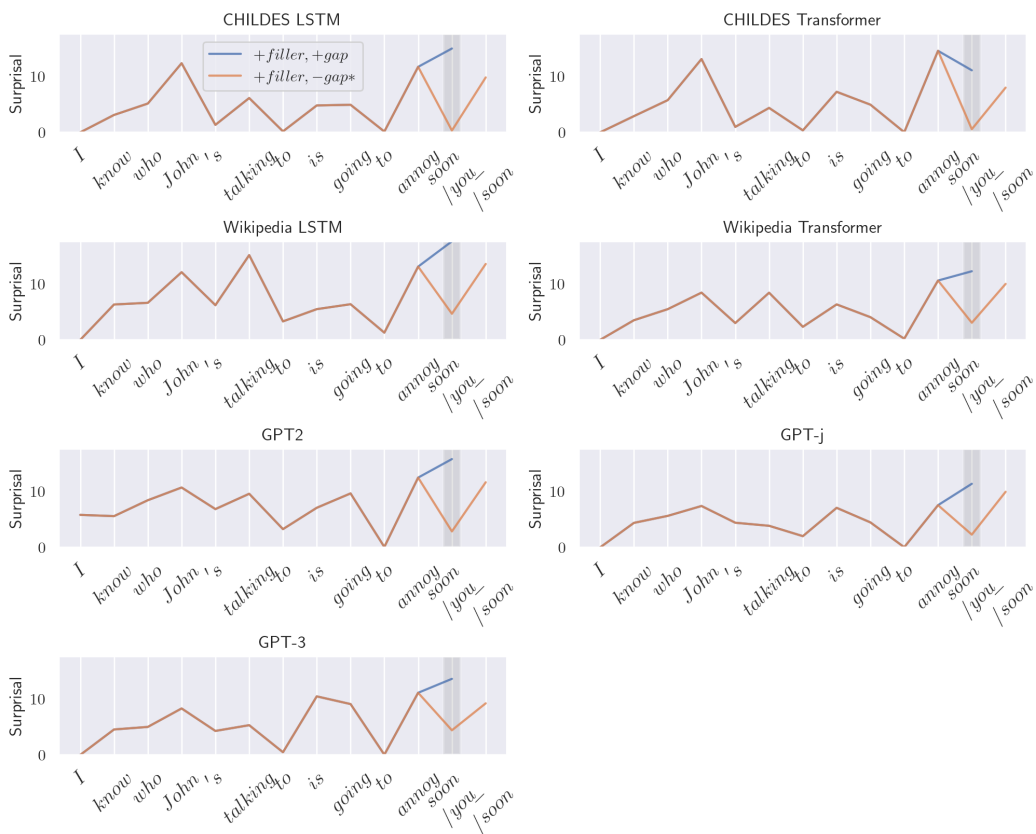


Figure 3: Raw surprisal values for the ungrammatical sentence (3a) which violates a subject island, in orange, and its grammatical variant (3b), in blue, where a parasitic gap makes it possible to escape the island. For measuring the model’s expectation for a gap, surprisal is measured at the adverb *soon*, which indicates a gap. This is compared with surprisal at *John* which plugs the gap at the same position. All networks wrongly assign a higher surprisal value to the grammatical continuation.

claim that LLMs undo the APS in this domain: for LLMs to undo this APS, they would need to provide a passable approximation of PG and ATB, but their performance above does not suggest such an approximation.

If our entire empirical basis is the failure we just saw, however, our conclusions will remain weak. This is so for the following reason: while the behavior of a good linguistically-neutral learner on the examples above would indeed be informative about the APS, it is possible that current ANNs are simply not sufficiently good

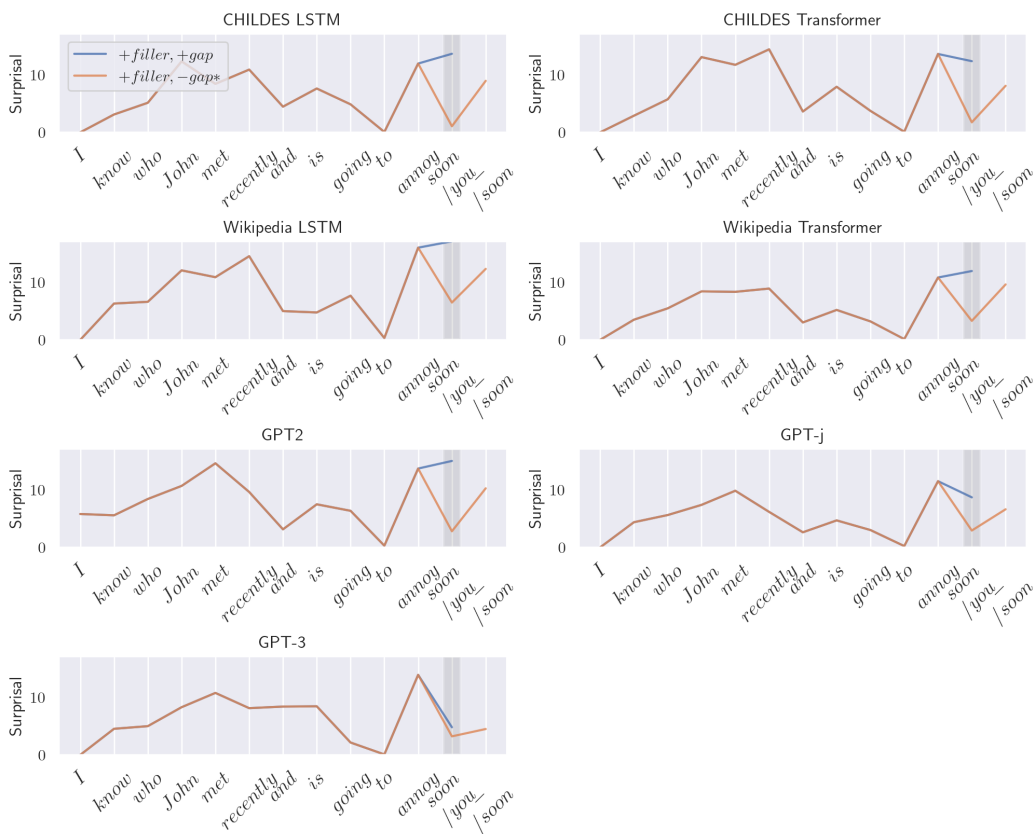


Figure 4: Raw surprisal values for the ungrammatical sentence (4a) which violates the coordinate structure constraint (orange), and its grammatical variant (4b) where ATB movement makes it possible to avoid the constraint (blue). All networks wrongly assign a higher surprisal value to the grammatical continuation *soon* rather than to *John*.

learners in this regard, and the inadequacies of the ANNs can in turn significantly limit our conclusions.

In the remainder of the present section we attempt to address the general concern about the adequacy of the ANNs, which we break down into two separate investigations. We first ask whether the failure that we just saw is an accident of the particular lexical choices that we used (Section 4.2). We then ask, building on WFL’s methodology, whether the failure was due to a general preference for ungapped continuations that is so strong as to override a preference for the correct form (Section 4.3). Our investigations concern ways in which the LLMs

might have an approximation of PG and ATB that is obscured by weaknesses of the models. By helping these LLMs at test we aim to reveal this approximation if it exists, but in both sections we will fail to find evidence for it. This, in turn, will strengthen the challenge to WFL’s claim: even with additional help at test, the models show no evidence that might undermine the APS.

4.2 Lexical Accident?

Our illustration above of how the LLMs prefer the ungrammatical continuation over the grammatical one for ATB and PG used one pair of sentences for each of the two patterns. This raises the obvious worry that the failure of the LLMs reflects some accidents of the specific sentences that we used. This worry is lessened to some extent by the fact that we looked at a broad range of different models trained on different corpora: it seems unlikely that all these models and all these training corpora just happen to have the same blind spot when it comes to the specific sentences that we used above and that otherwise the models approximate the patterns well. Still, it is clearly useful to examine more systematically what happens when we vary the lexical choices for the two patterns.

In order to test the performance of the networks on PG and ATB sentences more broadly, we systematically varied the lexical choices in (3) and (4), repeated here.

- (5) a. * I know who [John’s talking to __] is going to annoy you soon.
b. I know who [John’s talking to __] is going to annoy __ soon.
- (6) a. * I know who John [met __ recently] and [is going to annoy you soon].
b. I know who John [met __ recently] and [is going to annoy __ soon].

We generated the sentences by template, using simple context-free grammars. Excerpts from these grammars and a sample of the generated sentences are given in Tables 2 and 3. The full grammars are given in Appendix A.¹⁷ 8,064 sentence tuples were generated for PG and 6,624 for ATB. For a given model and a given pair of sentences, we looked at the surprisal of the model at the critical point on each member of the pair. For (5), for example, we checked whether after the shared prefix *I know who [John’s talking to __] is going to annoy ...* surprisal

¹⁷All experimental material and the source code are available at <https://github.com/Oxnurl/llm-poverty-of-stimulus>.

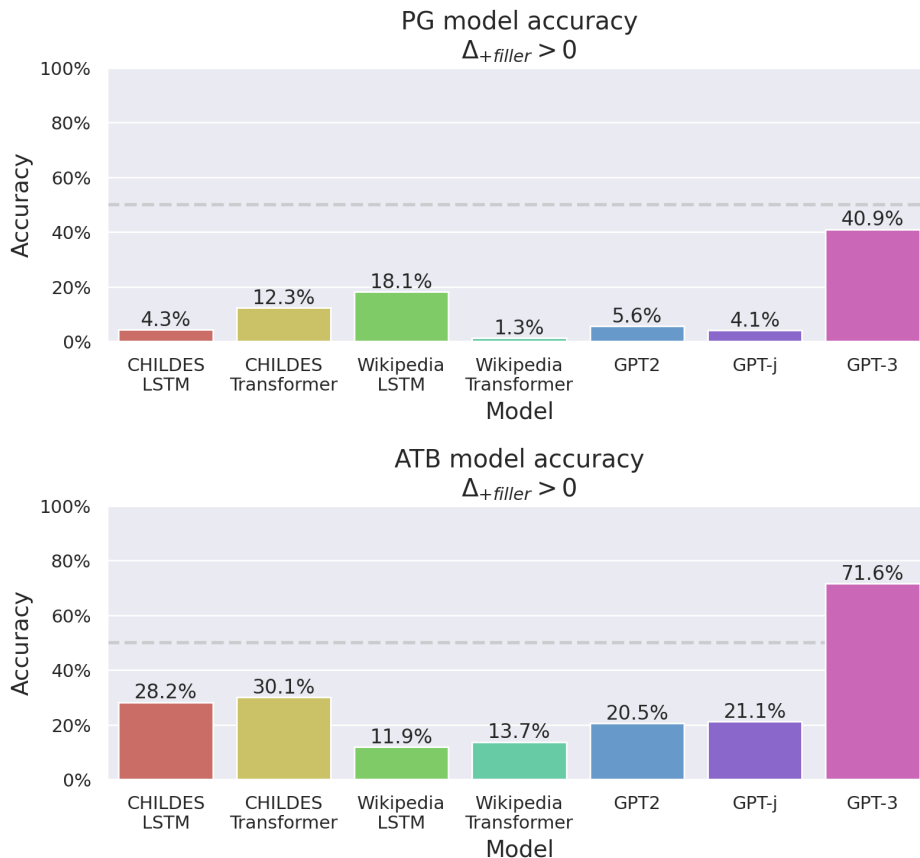


Figure 5: Model accuracy on the $\Delta_{+filler}$ condition for the PG and ATB datasets. Accuracy is measured as the ratio of cases where the model assigns a higher probability to the grammatical sentence continuation.

was higher at the ungapped, ungrammatical continuation *you* as in (5a) than in the gapped, grammatical continuation *soon* as in (5b). If it was — and in line with our lenient condition for success that is satisfied by any kind of preference for the grammatical continuation regardless of its magnitude — this counted as a success. We will write $\Delta = \text{Surprisal}(\text{ungapped continuation}|\text{shared prefix}) - \text{Surprisal}(\text{gapped continuation}|\text{shared prefix})$, and we will write $\Delta_{+filler}$ to indicate that the shared prefix has an upstream filler. Using this notation, we can write the condition for success as $\Delta_{+filler} > 0$.

Figure 5 plots the results of examining $\Delta_{+filler}$ preferences for the PG and ATB datasets. In both cases, the best performance by a large margin is that of

GPT-3, with 40.9% accuracy on the PG dataset and 71.6% accuracy on the ATB dataset. We are not sure to what extent these numbers can be taken to indicate an approximation of the relevant patterns by GPT-3. If it is a success then it is hardly a striking one. Nor is it particularly informative: recall that GPT-3 has been trained on the equivalent of 10,000 years of linguistic experience (and is also further improved manually in various ways), so even if it approximates the relevant patterns, this does not indicate that a general-purpose learner would acquire the relevant knowledge from a developmentally-realistic corpus of just a few years of linguistic experience. Setting GPT-3 aside, the models perform very poorly, with the best performance on PG being Wikipedia LSTM’s 18.1% accuracy and the best performance on ATB being CHILDES Transformer’s 30.1% accuracy. In other words, the models do not just fail to prefer the grammatical continuation over the ungrammatical one, they positively prefer the ungrammatical continuation in the vast majority of the pairs. Helping the LLMs by testing them on a wide range of lexical choices, then, fails to reveal any evidence that the models have approximated the patterns of parasitic gaps and across-the-board movement.

4.3 A Preference for Ungapped Continuations?

Our second investigation, building on WFL’s methodology, asks whether the networks have a local preference for or against gapped continuations that might make them succeed or fail for the wrong reasons.

Consider again (5) (=I know who [John’s talking to __] is going to annoy *you/✓ __ soon). A sufficiently strong local preference about the critical area can affect a given ANN’s success regardless of whether it has acquired any approximation of PG, or of wh-movement in general. It could be, for example, that the ANN assigns a higher probability to the grammatical continuation *soon* than to the ungrammatical *you* but that it does so because it ignores the filler (*who*) altogether and simply prefers *annoy soon* to *annoy you*. Conversely, it is conceivable that the ANN has, in fact, acquired knowledge of wh-movement but that it incorrectly prefers *you* to *soon* because of similarly irrelevant reasons. For example, perhaps it has a strong preference for ungapped continuations in general, or perhaps it has such a preference in the present case because the lexical frequency of *you* is very high.

To what extent might such local preferences affect the ANNs? We are not entirely sure. A good enough learner would presumably not get confused by such irrelevant factors, and the fact that all our models perform well on very simple filler-gap dependencies illustrated in Figures 1 and 2 is at least suggestive of their

ability to overcome any such confusion when the training data are sufficiently rich. However, beyond this suggestive evidence it is hard to tell whether current ANNs are good enough learners in this sense, and it strikes us as reasonable to further investigate possible confusion by irrelevant factors that might override the preference for the correct pattern.

Following WFL, we will explore the possible effect of irrelevant factors of the kind just mentioned by looking at each LLM’s preference for gapped over un-gapped continuations and comparing this preference when there is an upstream filler and when there is no such filler. When an upstream filler is present, the model’s preference for a gapped continuation (e.g., *annoy soon*) over the un-gapped continuation (*annoy you*) should be stronger than when an upstream filler is absent. In other words, we will be looking at whole paradigms of the shape we already saw in (1) and not just at those portions of the paradigm in which a filler is present. Such a paradigm is illustrated for PG in (7) and for ATB in (8). Underlined words indicate the \pm filler alternations. Words in bold indicate the critical region that shows whether the continuation is gapped or not.

(7) PG

	<i>+gap</i>	<i>-gap</i>
<i>+filler</i>	I know <u>who</u> John’s talking to is going to annoy soon .	*I know <u>who</u> John’s talking to is going to annoy you soon.
<i>-filler</i>	*I know <u>that</u> John’s talking to <u>Mary</u> is going to annoy soon .	I know <u>that</u> John’s talking to <u>Mary</u> is going to annoy you soon.

(8) ATB

	<i>+gap</i>	<i>-gap</i>
<i>+filler</i>	I know <u>who</u> John met recently and is going to annoy soon .	*I know <u>who</u> John met recently and is going to annoy you soon.
<i>-filler</i>	*I know <u>that</u> John met <u>Bob</u> recently and is going to annoy soon .	I know <u>that</u> John met <u>Bob</u> recently and is going to annoy you soon.

Extending our lenient condition for success used above, we will now consider it a success for a given model on a particular paradigm if its preference for the gapped continuation (regardless of its absolute magnitude or even its sign) is higher in the presence of an upstream filler than in its absence. Above we introduced the notation $\Delta = Surprisal(\text{ungapped continuation}|\text{shared prefix}) - Surprisal(\text{gapped}$

continuation|shared prefix) for the extent of the preference for the gapped continuation over the ungapped continuation, and we wrote $\Delta_{+filler}$ when the shared prefix had an upstream filler. We will now consider also the analogous $\Delta_{-filler}$, for the part of the paradigm where the shared prefix does not have an upstream filler. And we will consider it a success for the model if $\Delta_{+filler} > \Delta_{-filler}$. This lenient condition of cross-paradigm success follows the logic of difference-in-differences and is very much in line with WFL’s evaluation.¹⁸

In order to test the models across a large number of paradigms, with many different lexical choices, we used the same grammars mentioned in the previous section. In our earlier discussion we used the $+filler$ pairs generated by the grammar. In the present section we use also the corresponding $-filler$ pairs, and from each paradigm of $+filler$ and $-filler$ pairs we compute $\Delta_{\pm filler}$ values. Excerpts from the grammars are provided in Table 2 (for PGs) and Table 3 (for ATB).

Figure 6 plots the LLMs’ performance for the cross-paradigm (difference-in-differences) condition. All models except CHILDES LSTM have higher scores for the present measure of $\Delta_{+filler} > \Delta_{-filler}$ than they did for the earlier measure of $\Delta_{+filler} > 0$ (Figure 5), and this holds for both PG and ATB. However, we can see that only GPT-j and GPT-3 obtain scores that are convincingly high. But GPT-j is trained on the equivalent of 500 lifetimes of human linguistic exposure, and GPT-3 is trained on the equivalent of 100 lifetimes and fine-tuned further on downstream language tasks. Even GPT-2, trained on the equivalent of ten lifetimes — and thus two orders of magnitude at least above what a child hears by the time they have knowledge of PG and ATB — only reaches modest scores, below 80%. And the smaller models obtain much lower scores. This includes the English Wikipedia LSTM and Transformer, whose training corpus corresponds to about 8 years of linguistic exposure, arguably the most realistic developmentally in terms of size of all the models.

The gradual improvement of LLM scores as the corpora become very large suggests that current models are in principle capable of improving their approximation of the pattern of wh-movement, but also that this improvement requires much more information than what is present in a corpus that corresponds to anything a child might encounter. We return to the potential of richer training data to improve an LLMs approximation of the patterns under consideration in the next

¹⁸Of course, this new criterion still allows for various irrelevant factors to affect success. For example, a model could become successful simply by deciding that *who* corresponds to a high probability for *soon* and a low probability for *you* anywhere in the sentence and that *that* corresponds to the opposite. We set aside such worries here.

$S \rightarrow \langle \text{PREAMBLE} \rangle \langle \pm F \rangle \langle \pm G \rangle$
 $\langle \text{PREAMBLE} \rangle \rightarrow I \text{ know}$
 $\langle +F \rangle \rightarrow \underline{who} \langle \text{NAME1} \rangle \langle \text{GEN} \rangle \langle \text{NP} \rangle$
 $\langle -F \rangle \rightarrow \underline{that} \langle \text{NAME1} \rangle \langle \text{GEN} \rangle \langle \text{NP} \rangle \langle \underline{\text{NAME2}} \rangle$
 $\langle +G \rangle \rightarrow \langle \text{LINK} \rangle \langle \text{V} \rangle \langle \mathbf{ADV} \rangle$
 $\langle -G \rangle \rightarrow \langle \text{LINK} \rangle \langle \text{V} \rangle \langle \mathbf{OBJ} \rangle \langle \text{ADV} \rangle$
 $\langle \text{GEN} \rangle \rightarrow 's$
 $\langle \text{NP} \rangle \rightarrow \langle \text{NP_SIMPLE} \rangle \mid \langle \text{NP_COMPLEX} \rangle$
 $\langle \text{NP_SIMPLE} \rangle \rightarrow \langle \text{GERUND} \rangle$
 $\langle \text{NP_COMPLEX} \rangle \rightarrow \langle \text{N_EMBEDDED} \rangle \text{ 'to' } \langle \text{V_EMBEDDED} \rangle$
 $\langle \text{LINK} \rangle \rightarrow \text{'is about to' } \mid \text{'is likely to' } \mid \text{'is going to' } \mid \text{'is expected to'}$
 $\langle \text{V} \rangle \rightarrow \text{'bother' } \mid \text{'annoy' } \mid \text{'disturb'}$
 $\langle \text{OBJ} \rangle \rightarrow \text{'you' } \mid \text{'us' } \mid \text{'Kim'}$
 $\langle \text{GERUND} \rangle \rightarrow \text{'talking to' } \mid \text{'dancing with' } \mid \text{'playing with'}$
 $\langle \text{N_EMBEDDED} \rangle \rightarrow \text{'decision' } \mid \text{'intent' } \mid \text{'effort' } \mid \text{'attempt' } \mid \text{'failure'}$
 $\langle \text{V_EMBEDDED} \rangle \rightarrow \text{'talk to' } \mid \text{'call' } \mid \text{'meet' } \mid \text{'dance with' } \mid \text{'play with'}$
 $\langle \text{ADV} \rangle \rightarrow \text{'soon' } \mid \text{'eventually'}$
...
 $\Rightarrow I \text{ know } \underline{who} \text{ John's talking to is going to annoy } \mathbf{soon}. (+\text{filler}, +\text{gap})$
 $\Rightarrow * I \text{ know } \underline{who} \text{ John's talking to is going to annoy } \mathbf{you} \text{ soon}. (+\text{filler}, -\text{gap})$
 $\Rightarrow * I \text{ know } \underline{that} \text{ John's talking to } \underline{Mary} \text{ is going to annoy } \mathbf{soon}. (-\text{filler}, +\text{gap})$
 $\Rightarrow I \text{ know } \underline{that} \text{ John's talking to } \underline{Mary} \text{ is going to annoy } \mathbf{you} \text{ soon}. (-\text{filler}, -\text{gap})$

Table 2: Excerpt from the context-free grammar used to generate PG sentences for the experiments in Section 4.3, and sample sentences generated from it. Underlined words alternate according to the $\pm \text{filler}$ condition; words in bold mark the position where the $\pm \text{gap}$ condition becomes evident and surprisal is measured.

section. In the meantime we conclude that even with considerable help at test, the performance of the LLMs provides no evidence against the APS.

$S \rightarrow \langle \text{PREAMBLE} \rangle \langle \pm F \rangle \langle \text{LINK} \rangle \langle \pm G \rangle$
 $\langle \text{PREAMBLE} \rangle \rightarrow I \text{ know}$
 $\langle +F \rangle \rightarrow \underline{\text{who}} \langle \text{NAME1} \rangle \langle \text{VP1} \rangle \langle \text{ADV1} \rangle$
 $\langle -F \rangle \rightarrow \underline{\text{that}} \langle \text{NAME1} \rangle \langle \text{VP1} \rangle \langle \underline{\text{NAME2}} \rangle \langle \text{ADV1} \rangle$
 $\langle +G \rangle \rightarrow \langle \text{LINK} \rangle \langle \text{VP2} \rangle \langle \mathbf{ADV2} \rangle$
 $\langle -G \rangle \rightarrow \langle \text{LINK} \rangle \langle \text{VP2} \rangle \langle \mathbf{OBJ} \rangle \langle \text{ADV2} \rangle$
 $\langle \text{LINK} \rangle \rightarrow \text{'and is going to'}$
 $\langle \text{ADV1} \rangle \rightarrow \text{'recently' | 'lately'}$
 $\langle \text{ADV2} \rangle \rightarrow \text{'soon' | 'today' | 'now'}$
 $\langle \text{VP1} \rangle \rightarrow \langle \text{VP1_SIMPLE} \rangle \mid \langle \text{VP1_COMPLEX} \rangle$
 $\langle \text{VP1_SIMPLE} \rangle \rightarrow \text{'met' | 'saw'}$
 $\langle \text{VP2} \rangle \rightarrow \langle \text{VP2_SIMPLE} \rangle \mid \langle \text{VP2_COMPLEX} \rangle$
 $\langle \text{VP2_SIMPLE} \rangle \rightarrow \text{'hug' | 'slap' | 'kiss'}$
 $\langle \text{OBJ} \rangle \rightarrow \text{'you' | 'us' | 'Kim'}$
 ...
 $\Rightarrow I \text{ know } \underline{\text{who}} \text{ John met recently and is going to hug } \mathbf{\text{soon}}. (+\text{filler}, +\text{gap})$
 $\Rightarrow *I \text{ know } \underline{\text{who}} \text{ John met recently and is going to hug } \mathbf{\text{you}} \text{ soon}. (+\text{filler}, -\text{gap})$
 $\Rightarrow *I \text{ know } \underline{\text{that}} \text{ John met } \underline{\text{Bob}} \text{ recently and is going to hug } \mathbf{\text{soon}}. (-\text{filler}, +\text{gap})$
 $\Rightarrow I \text{ know } \underline{\text{that}} \text{ John met } \underline{\text{Bob}} \text{ recently and is going to hug } \mathbf{\text{you}} \text{ soon}. (-\text{filler}, -\text{gap})$

Table 3: Excerpt from the context-free grammar used to generate ATB sentences for the experiments in Section 4.3, and sample sentences generated from it. Underlined words alternate according to the $\pm \text{filler}$ condition; words in bold mark the position where the $\pm \text{gap}$ condition becomes evident and surprisal is measured.

5 A General Inability to Acquire a Suitable Preference?

Recall WFL’s contention that LLMs show that a linguistically-neutral learner can acquire knowledge of wh-movement from a realistic corpus. In the face of our results from the previous section, WFL’s claim needs to be abandoned: current LLMs provide no basis for such a conclusion. Of course, this is not the same as saying that LLMs provide evidence *for* the APS: the failure of the LLMs might be

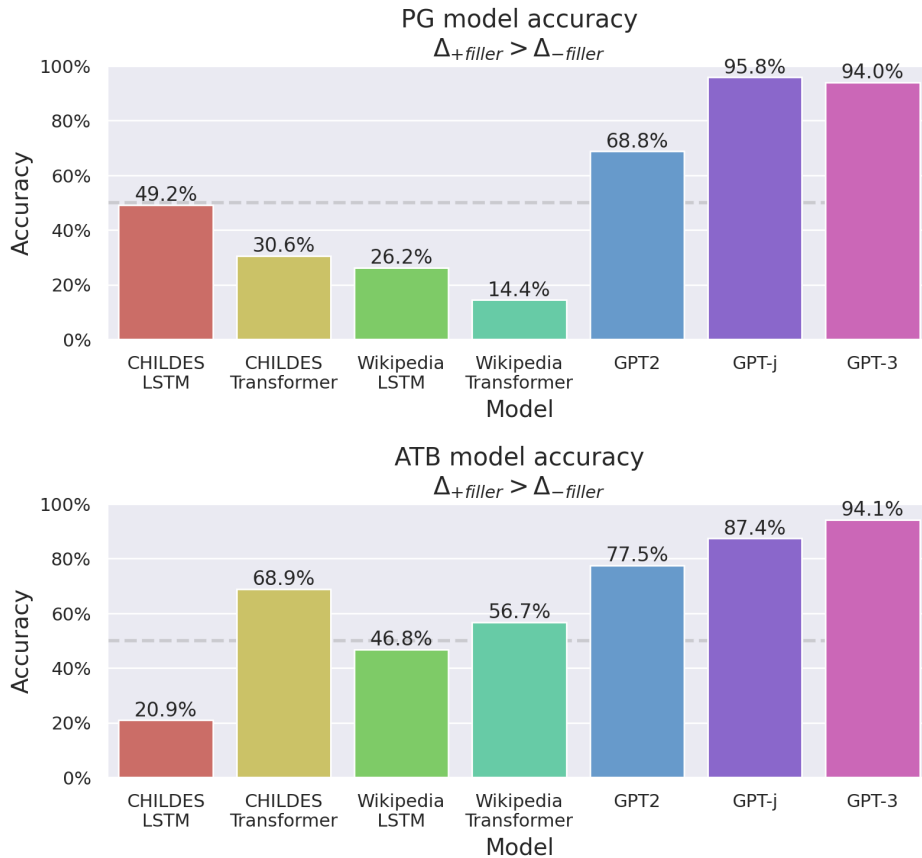


Figure 6: Model accuracy on the difference-in-differences condition for the PG and ATB datasets. Accuracy is measured as the ratio of cases where $\Delta_{+filler} > \Delta_{-filler}$, that is, when the model shows a relative higher preference for a gap when the gap follows a filler than when it does not.

due entirely to their own limitations and not be informative about the richness of the training corpora. In the present section, however, we will go one step further and provide tentative evidence that the failure of the LLMs is due also to the insufficient richness of the training corpora and not just to weaknesses of the ANNs. We do so by helping one of our models at training: we retrain the Wikipedia Transformer model on an enriched corpus that includes multiple instances of PG and ATB. As we show, the performance of the model improves significantly on the enriched corpus, suggesting that the failure on the original corpus reflects the poverty of that corpus.

The additional instances for the enriched training corpus are generated by template, using a variant of the CFGs that we used in sections 4.2 and 4.3. To increase the probability that an improved performance by the model will reflect generalization rather than memorization, the structure of the additional instances is different from that of the test sentences from Section 4.3. Specifically, we made the additional sentences slightly simpler than the test sentences, focusing on the transitive verb whose object position forms the second gap (the main gap in the PG examples and the second-conjunct gap in the ATB examples): while in the test sentences the relevant transitive verb is always embedded under at least a raising predicate (e.g., *likely*) and sometimes under additional clauses, in the additional training sentences such embedding is absent. Example training and test sentences are given in Table 4. The full CFGs used in creating the additional instances are given in Appendix C.

From each CFG of each phenomenon (PG/ATB), we sampled 100 sentences for the two grammatical conditions ($+filler, +gap$ and $-filler, -gap$), totaling 200 extra sentences. These sentences were added to the original English Wikipedia dataset, and the model was trained using the same regime as in Yedetore et al. (2023) (itself based on the training regime in Gulordava et al., 2018). The model was trained for 48 hours or until reaching the early-stop condition from Yedetore et al. (2023) which stops the training if the validation loss does not improve for more than two consecutive epochs. Due to the long training times of the model, the results reported here are for one random seed with no hyper-parameter search. Since the goal of this experiment was to demonstrate the model’s ability to improve significantly given more data, this was sufficient.

The model’s performance on the training and test set, before and after retraining, is visualized in Figure 7.

For both ATB and PG the performance of the model improves significantly. For ATB, the raw $\Delta_{+filler} > 0$ accuracy score improves from 13.7% to 35.8%, and the difference-in-differences $\Delta_{+filler} > \Delta_{-filler}$ score improves from 56.7% to 97.2%. This is a dramatic improvement over the performance of the model on its original corpus and is higher than the performance of other architectures when trained on a much larger corpus. For PG, the raw score improves modestly, from 1.3% to 5.3%, while the difference-in-differences score improves from 14.4% to 65.5%. The raw score for PG certainly doesn’t inspire confidence that the model has acquired the dependency. Recall, however, that this is not what we were after here. Our question was whether the model is so weak that its poor performance when trained on the original corpus reflects its inability to do better. The retraining results show that the model can do considerably better once the corpus is

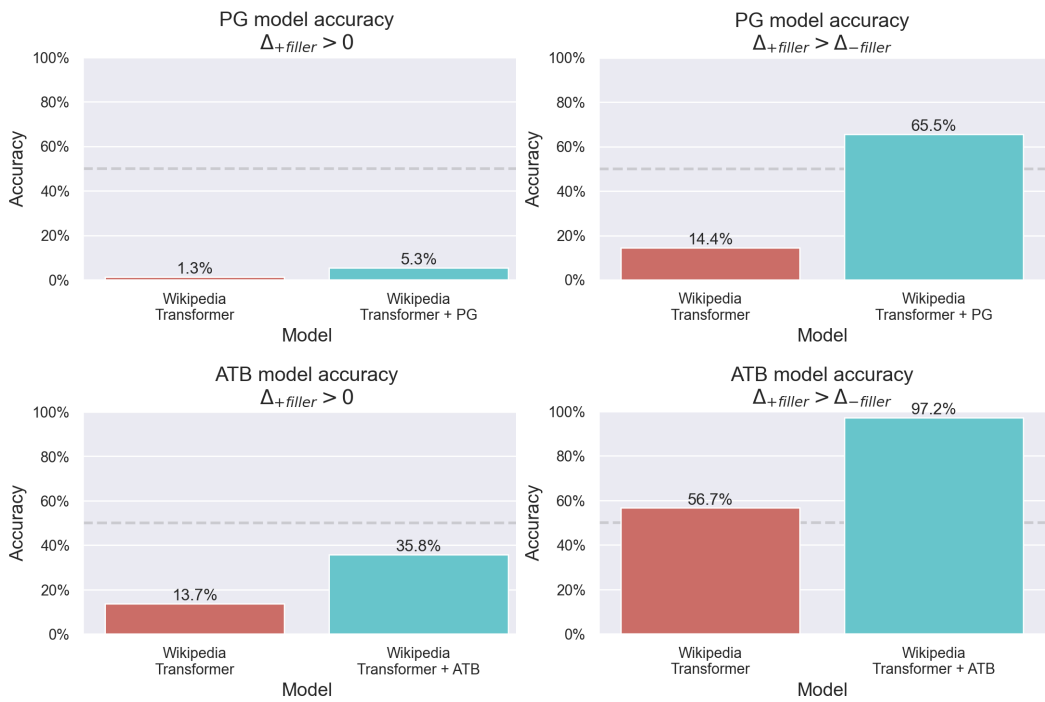


Figure 7: Accuracy for the retrained Transformer model, when trained on the original Wikipedia vs. when trained on the same dataset with extra PG and ATB sentences. The left figures plot accuracy for the $+filler$ condition, and the right figures plot accuracy for the difference-in-difference condition.

PG	Training Examples
	<ul style="list-style-type: none"> • I know who John’s attitude towards upset yesterday. • I know who John’s friendship with will annoy soon. • I know who John’s praising of amused lately.
	Test Examples
	<ul style="list-style-type: none"> • I know who John’s talking to is about to bother soon. • I know who John’s playing with is going to annoy eventually. • I know who John’s failure to dance with is going to disturb soon.
ATB	Training Examples
	<ul style="list-style-type: none"> • I know who John saw yesterday and kissed today. • I know who John helped recently and married today. • I know who John hugged often and will insult soon.
	Test Examples
	<ul style="list-style-type: none"> • I know who John met recently and is going to complain to Patricia about soon. • I know who John said that Mary saw lately and is going to be glad to hug now. • I know who John asked Mary about lately and is going to claim that Patricia will hug today.

Table 4: Example training and test sentences for the retraining task in Section 5. A sample of simplified training PG and ATB sentences are added to the model’s original training data (English Wikipedia), and the model is then tested on the full battery of sentences from Section 4.3. The full CFGs for the training and test datasets are given in Appendices A and C.

sufficiently rich.

Caution is required in interpreting this result. Like all current LLMs, our model is opaque, and we are limited in the conclusions that we can draw from it. In particular, while we observe that the model’s performance improved when retrained on a corpus that is enriched in a certain way, it is possible that there are other kinds of evidence for the patterns under consideration that a good general-purpose learner would be able to make use of and that our model cannot. What we found is consistent with such evidence being in the original corpus. Our use of retraining data that were structurally different from the test data was aimed at lessening this worry, since improvement suggests an ability to generalize and not just memorize. This, in turn, increases the plausibility that the model would

have been able to generalize from other kinds of relevant examples if the original corpus had been sufficiently rich. But the opacity of the model prevents us from saying more, and our results here must be qualified accordingly.

6 Conclusion

The APS has been central to linguists’ reasoning about innateness for a long time. It has always been difficult, however, to estimate just how much information a linguistically-neutral learner might hope to extract from a realistic input. Modern ANNs promise to change this, and their linguistic knowledge and learning have been the topic of research of a growing literature. We focused here on WFL, who use LLMs to argue that the stimulus is rich enough when it comes to wh-movement and that this dismantles the APS in this domain. We showed that this conclusion is premature: by looking at parasitic gaps and across-the-board movement we showed that several ANNs fail to reach a passable approximation of the pattern of wh-movement.

Is it possible that some future linguistically-neutral learner will succeed where the models that we have examined have failed? Of course. As we mentioned, current models are too opaque and too poorly understood (and current training corpora are too unrealistic developmentally) to definitively settle the question of whether the APS for wh-movement holds. We note, however, that the architectures we have considered are generally successful in approximating many other aspects of linguistic data and that we evaluated the models using an extremely lenient criterion for success. And some of the models have been provided with very generous amounts of linguistic input, in some cases several orders of magnitude beyond what children receive. Given that none of the ANNs reached an adequate approximation of the pattern for the relatively simple examples that we have considered — and given that at least one network did seem capable of improving its approximation when retrained on a clearly rich corpus — we find it likelier that the stimulus is simply too poor to warrant the acquisition of the relevant aspects of knowledge from a corpus that is even remotely realistic developmentally by a linguistically-neutral learner. And if that turns out to be the case, adult speakers’ knowledge of these aspects is evidence that children are innately endowed in ways that are not linguistically neutral.

References

- Baker, C. L. 1978. *Introduction to generative transformational syntax*. Englewood Cliffs, New Jersey: Prentice-Hall.
- Bernardy, Jean-Philippe, and Shalom Lappin. 2017. Using deep neural networks to learn syntactic agreement. *LiLT (Linguistic Issues in Language Technology)* 15.
- Berwick, Robert C. 2018. Revolutionary new ideas appear infrequently. In *Syntactic Structures after 60 years: The impact of the Chomskyan revolution in linguistics*, ed. Norbert Hornstein, Howard Lasnik, Pritty Patel-Grosz, and Charles Yang, volume 60, 177–194. Walter de Gruyter.
- Berwick, Robert C., Paul Pietroski, Beracah Yankama, and Noam Chomsky. 2011. Poverty of the stimulus revisited. *Cognitive Science* 35:1207–1242.
- Bhattacharya, Debasmita, and Marten van Schijndel. 2020. Filler-gaps that neural networks fail to generalize. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, ed. Raquel Fernández and Tal Linzen, 486–495. Online: Association for Computational Linguistics.
- Bock, Kathryn, and Carol A Miller. 1991. Broken agreement. *Cognitive Psychology* 23:45–93.
- Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33:1877–1901.
- Chaves, Rui P. 2020. What don't RNN language models learn about filler-gap dependencies? *Proceedings of the Society for Computation in Linguistics* 3:20–30.
- Chomsky, Noam. 1957. *Syntactic structures*. The Hague: Mouton.
- Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.
- Chomsky, Noam. 1971. *Problems of knowledge and freedom: The Russell lectures*. Pantheon Books.
- Chomsky, Noam. 1975. *Current issues in linguistic theory*. The Hague: Mouton.
- Chomsky, Noam. 1980. *Rules and representations*. New York: Columbia University Press.
- Chowdhury, Shammur Absar, and Roberto Zamparelli. 2018. RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, ed. Emily M. Bender, Leon Derczynski, and Pierre Isabelle, 133–144. Santa Fe, New Mexico,

- USA: Association for Computational Linguistics.
- Dyer, Fred C., and Jeffrey A. Dickinson. 1994. Development of sun compensation by honeybees: How partially experienced bees estimate the sun’s course. *Proceedings of the National Academy of Sciences* 91:4471–4474.
- El-Naggar, Nadine, Andrew Ryzhikov, Laure Daviaud, Pranava Madhyastha, and Tillman Weyde. 2023. Formal and empirical studies of counting behaviour in relu rnns. In *Proceedings of 16th edition of the International Conference on Grammatical Inference*, ed. François Coste, Faissal Ouardi, and Guillaume Rabusseau, volume 217 of *Proceedings of Machine Learning Research*, 199–222. PMLR.
- Elman, Jeffrey L., Elizabeth Bates, M. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. 1996. *Rethinking innateness: A connectionist perspective on development*. Cambridge, MA: The MIT Press.
- Engdahl, E. 1983. Parasitic gaps. *Linguistics and Philosophy* 6:5–34.
- Foraker, Stephani, Terry Regier, Naveen Khetarpal, Amy Perfors, and Joshua Tenenbaum. 2009. Indirect evidence and the poverty of the stimulus: The case of anaphoric one. *Cognitive Science* 33:287–300.
- Fox, Danny. 2000. *Economy and semantic interpretation*. Cambridge, MA: MIT Press.
- Gordon, Peter. 1985. Level-ordering in lexical development. *Cognition* 21:73–93.
- Gulordava, Kristina, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL 2018*, 1195–1205.
- Haïk, Isabelle. 1985. The syntax of operators. Doctoral Dissertation, MIT.
- Hart, Betty, and Todd R. Risley. 1995. *Meaningful differences in the everyday experience of young American children*. Baltimore: Paul H. Brookes.
- Hornstein, Norbert, and Jairo Nunes. 2002. On asymmetries between parasitic gap and across-the-board constructions. *Syntax* 5.
- Hsu, Anne S., and Nick Chater. 2010. The logical problem of language acquisition: A probabilistic perspective. *Cognitive Science* 34:972–1016.
- Huebner, Philip A., Elior Sulem, Fisher Cynthia, and Dan Roth. 2021. Baby-BERTa: Learning more grammar with small-scale child-directed language. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, ed. Arianna Bisazza and Omri Abend, 624–646. Online: Association for Computational Linguistics.
- Jozefowicz, Rafal, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *arXiv:1602.02410 [cs]*.

- Katzir, Roni. 2023. Why large language models are poor theories of human linguistic cognition: A reply to Piantadosi. *Biolinguistics* 17.
- Kodner, Jordan, and Nitish Gupta. 2020. Overestimation of syntactic representation in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1757–1762. Online: Association for Computational Linguistics.
- Kodner, Jordan, Sarah Payne, and Jeffrey Heinz. 2023. Why linguistics will thrive in the 21st century: A reply to Piantadosi (2023). ArXiv preprint arXiv:2308.03228.
- Kuncoro, Adhiguna, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1426–1436.
- Lan, Nur, Emmanuel Chemla, and Roni Katzir. 2023. Benchmarking neural network generalization for grammar induction. In *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*, 131–140. Gothenburg, Sweden: Association for Computational Linguistics.
- Lan, Nur, Emmanuel Chemla, and Roni Katzir. 2024. Bridging the empirical-theoretical gap in neural network formal language learning using minimum description length. Submitted. arXiv preprint arXiv:2402.10013, February 2024.
- Lan, Nur, Michal Geyer, Emmanuel Chemla, and Roni Katzir. 2022. Minimum description length recurrent neural networks. *Transactions of the Association for Computational Linguistics* 10:785–799.
- Legate, Julie Anne, and Charles Yang. 2002. Empirical re-assessment of stimulus poverty arguments. *The Linguistic Review* 19.
- Lewis, John D, and Jeffery L Elman. 2001. A connectionist investigation of linguistic arguments from the poverty of the stimulus: Learning the unlearnable. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 23.
- Lidz, Jeffrey, Sandra Waxman, and Jennifer Freedman. 2003. What infants know about syntax but couldn't have learned: Experimental evidence for syntactic structure at 18 months. *Cognition* 89:B65–B73.
- Linzen, Tal, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4:521–535.
- MacWhinney, Brian. 2014. *The CHILDES Project: Tools for Analyzing Talk, Volume II: The Database*. New York: Psychology Press, 3 edition.

- Marvin, Rebecca, and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, ed. Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, 1192–1202. Brussels, Belgium: Association for Computational Linguistics.
- Merrill, William, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. A formal hierarchy of RNN architectures. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ed. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, 443–459. Online: Association for Computational Linguistics.
- Moro, Andrea, Matteo Greco, and Stefano F. Cappa. 2023. Large languages, impossible languages and human brains. *Cortex* 167:82–85.
- Munn, Alan. 1992. A null operator analysis of ATB gaps. *The Linguistic Review* 9:1–26.
- Nissenbaum, Jon. 2000. Investigations of covert phrase movement. Doctoral Dissertation, MIT, Cambridge, Mass.
- Ozaki, Satoru, Dan Yurovsky, and Lori Levin. 2022. How well do LSTM language models learn filler-gap dependencies? *Proceedings of the Society for Computation in Linguistics* 5:76–88.
- Pearl, Lisa, and Jon Sprouse. 2013. Syntactic islands and learning biases: Combining experimental syntax and computational modeling to investigate the language acquisition problem. *Language Acquisition* 20:23–68.
- Perfors, Amy, Joshua Tenenbaum, and Terry Regier. 2011. The learnability of abstract syntactic principles. *Cognition* 118:306–338.
- Phillips, Colin. 2013. On the nature of island constraints II: Language learning and innateness. In *Experimental syntax and island effects*, ed. Jon Sprouse and Norbert Hornstein, 132–157. Cambridge University Press.
- Piantadosi, Steven T. 2023. Modern language models refute Chomsky's approach to language. Ms., available at <https://ling.auf.net/lingbuzz/007180>, March 2023.
- Postal, Paul M. 1993. Parasitic gaps and the across-the-board phenomenon. *Linguistic Inquiry* 24:735–754.
- Pullum, Geoffrey, and Barbara Scholz. 2002. Empirical assessment of stimulus poverty arguments. *The Linguistic Review* 19:9–50.
- Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1:9.
- Rawski, Jon, and J Baumont. 2023. Modern language models refute nothing. *Lingbuzz Preprint* .

- Rawski, Jonathan, and Jeffrey Heinz. 2019. No free lunch in linguistics or machine learning: Response to pater. *Language* 95:e125–e135.
- Reali, Florencia, and Morten Christiansen. 2005. Uncovering the richness of the stimulus: Structure dependence and indirect statistical evidence. *Cognitive Science* 29:1007–1028.
- Ross, John R. 1967. Constraints on variables in syntax. Doctoral Dissertation, MIT, Cambridge, MA.
- Siegelmann, H. T., and E. D. Sontag. 1995. On the computational power of neural nets. *Journal of Computer and System Sciences* 50:132–150.
- Siegelmann, Hava T., and Eduardo D. Sontag. 1991. Turing computability with neural nets. *Applied Mathematics Letters* 4:77–80.
- Sprouse, Jon, Beracah Yankama, Sagar Indurkha, Sandiway Fong, and Robert C. Berwick. 2018. Colorless green ideas do sleep furiously: gradient acceptability and the nature of the grammar. *The Linguistic Review* 35:575–599.
- Strobl, Lena, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2023. Transformers as Recognizers of Formal Languages: A Survey on Expressivity. *arXiv preprint arXiv:2311.00208*.
- Vázquez Martínez, Héctor, Annika Lea Heuser, Charles Yang, and Jordan Kodner. 2023. Evaluating neural language models as cognitive models of language acquisition. In *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, ed. Dieuwke Hupkes, Verna Dankers, Khuyagbaatar Batsuren, Koustuv Sinha, Amirhossein Kazemnejad, Christos Christodoulopoulos, Ryan Cotterell, and Elia Bruni, 48–64. Singapore: Association for Computational Linguistics.
- Wagers, M.W., E.F. Lau, and C. Phillips. 2009. Agreement attraction in comprehension: representations and processes. *Journal of Memory and Language* 61:206–237.
- Wang, Ben, and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model. URL <https://github.com/kingoflolz/mesh-transformer-jax>.
- Warstadt, Alex, and Samuel R Bowman. 2022. What artificial neural networks can tell us about human language acquisition. In *Algebraic structures in natural language*, 17–60. CRC Press.
- Warstadt, Alex, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics* 8:377–392.
- Weiss, Gail, Yoav Goldberg, and Eran Yahav. 2018. On the practical computa-

- tional power of finite precision rnns for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 740–745. Melbourne, Australia: Association for Computational Linguistics.
- Wilcox, Ethan, Roger Levy, and Richard Futrell. 2019. What syntactic structures block dependencies in RNN language models? *arXiv preprint arXiv:1905.10431*.
- Wilcox, Ethan, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about filler-gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 211–221.
- Wilcox, Ethan, Ethan Gotlieb, Richard Futrell, and Roger Levy. 2023. Using computational models to test syntactic learnability. *Linguistic Inquiry* 1–44.
- Williams, Edwin. 1990. The ATB-theory of parasitic gaps. *The Linguistic Review* 6:265–279.
- Williams, Edwin S. 1977. Across-the-board application of rules. *Linguistic Inquiry* 8:419–423.
- Wilson, Colin. 2006. Learning phonology with substantive bias: An experimental and computational study of velar palatalization. *Cognitive Science* 30:945–982.
- Yedetore, Aditya, Tal Linzen, Robert Frank, and R. Thomas McCoy. 2023. How poor is the stimulus? evaluating hierarchical generalization in neural networks trained on child-directed speech. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ed. Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, 9370–9393. Toronto, Canada: Association for Computational Linguistics.

A Appendix: Context-Free Grammars

A.1 PG

$$\begin{aligned}
 \langle S \rangle &\rightarrow \langle S_FG \rangle \\
 \langle S_FG \rangle &\rightarrow \langle PREAMBLE \rangle \langle F \rangle \langle G \rangle \\
 \langle S_XG \rangle &\rightarrow \langle UNGRAMMATICAL \rangle \langle PREAMBLE \rangle \langle XF \rangle \langle G \rangle \\
 \langle S_FX \rangle &\rightarrow \langle UNGRAMMATICAL \rangle \langle PREAMBLE \rangle \langle F \rangle \langle XG \rangle \\
 \langle S_XX \rangle &\rightarrow \langle PREAMBLE \rangle \langle XF \rangle \langle XG \rangle \\
 \langle UNGRAMMATICAL \rangle &\rightarrow '*'
 \end{aligned}$$

⟨GEN⟩ → ‘s’
 ⟨OBJ⟩ → ‘you’ | ‘us’ | ‘Kim’
 ⟨NAME1⟩ → ‘Bob’ | ‘John’
 ⟨NAME2⟩ → ‘Mary’ | ‘Jennifer’
 ⟨NAME3⟩ → ‘James’ | ‘Michael’
 ⟨NAME4⟩ → ‘Patricia’ | ‘Linda’
 ⟨PREAMBLE⟩ → ‘I know’
 ⟨F⟩ → ‘who’ ⟨NAME1⟩ ⟨GEN⟩ ⟨NP⟩
 ⟨XF⟩ → ‘that’ ⟨NAME1⟩ ⟨GEN⟩ ⟨NP⟩ ⟨NAME2⟩
 ⟨G⟩ → ⟨LINK⟩ ⟨V⟩ ⟨ADV2⟩
 ⟨XG⟩ → ⟨LINK⟩ ⟨V⟩ ⟨OBJ⟩ ⟨ADV2⟩
 ⟨NP⟩ → ⟨NP_SIMPLE⟩ | ⟨NP_COMPLEX⟩
 ⟨NP_SIMPLE⟩ → ⟨GERUND⟩
 ⟨NP_COMPLEX⟩ → ⟨N_EMBEDDED⟩ ‘to’ ⟨V_EMBEDDED⟩
 ⟨LINK⟩ → ‘is about to’ | ‘is likely to’ | ‘is going to’ | ‘is expected to’
 ⟨V⟩ → ‘bother’ | ‘annoy’ | ‘disturb’
 ⟨GERUND⟩ → ‘talking to’ | ‘dancing with’ | ‘playing with’
 ⟨N_EMBEDDED⟩ → ‘decision’ | ‘intent’ | ‘effort’ | ‘attempt’ | ‘failure’
 ⟨V_EMBEDDED⟩ → ‘talk to’ | ‘call’ | ‘meet’ | ‘dance with’ | ‘play with’
 ⟨ADV1⟩ → ‘recently’ | ‘earlier’
 ⟨ADV2⟩ → ‘soon’ | ‘eventually’

A.2 ATB

⟨S⟩ → ⟨S_FG⟩
 ⟨S_FG⟩ → ⟨PREAMBLE⟩ ⟨F⟩ ⟨G⟩
 ⟨S_XG⟩ → ⟨UNGRAMMATICAL⟩ ⟨PREAMBLE⟩ ⟨XF⟩ ⟨G⟩
 ⟨S_FX⟩ → ⟨UNGRAMMATICAL⟩ ⟨PREAMBLE⟩ ⟨F⟩ ⟨XG⟩
 ⟨S_XX⟩ → ⟨PREAMBLE⟩ ⟨XF⟩ ⟨XG⟩
 ⟨UNGRAMMATICAL⟩ → ‘*’
 ⟨GEN⟩ → ‘s’
 ⟨OBJ⟩ → ‘you’ | ‘us’ | ‘Kim’
 ⟨NAME1⟩ → ‘John’
 ⟨NAME2⟩ → ‘Mary’
 ⟨NAME3⟩ → ‘Bob’
 ⟨NAME4⟩ → ‘Patricia’
 ⟨PREAMBLE⟩ → ‘I know’

⟨F⟩ → ‘who’ ⟨NAME1⟩ ⟨VP1⟩ ⟨ADV1⟩
 ⟨XF⟩ → ‘that’ ⟨NAME1⟩ ⟨VP1⟩ ⟨NAME3⟩ ⟨ADV1⟩
 ⟨G⟩ → ⟨LINK⟩ ⟨VP2⟩ ⟨ADV2⟩
 ⟨XG⟩ → ⟨LINK⟩ ⟨VP2⟩ ⟨OBJ⟩ ⟨ADV2⟩
 ⟨ADV1⟩ → ‘recently’ | ‘lately’
 ⟨ADV2⟩ → ‘soon’ | ‘today’ | ‘now’
 ⟨LINK⟩ → ‘and is going to’
 ⟨VP1⟩ → ⟨VP1_SIMPLE⟩ | ⟨VP1_COMPLEX⟩
 ⟨VP1_SIMPLE⟩ → ‘met’ | ‘saw’
 ⟨VP1_COMPLEX⟩ → ⟨VP1_ABOUT⟩ | ⟨VP1_TO⟩ | ⟨VP1_ADJ⟩ | ⟨VP1_EMBEDDED⟩
 ⟨VP1_ABOUT⟩ → ⟨V_ABOUT_PAST⟩ ⟨NAME2⟩ ‘about’
 ⟨V_ABOUT_PAST⟩ → ‘asked’ | ‘told’
 ⟨VP1_TO⟩ → ⟨V_TO_PAST⟩ ⟨NAME2⟩ ‘to’ ⟨V_TRANS_INF_TO⟩
 ⟨V_TO_PAST⟩ → ‘wanted’ | ‘asked’
 ⟨V_TRANS_INF_TO⟩ → ‘call’ | ‘invite’
 ⟨VP1_ADJ⟩ → ‘was’ ⟨ADJ1⟩ ‘to’ ⟨V_TRANS_INF_ADJ⟩
 ⟨ADJ1⟩ → ‘eager’ | ‘happy’
 ⟨V_TRANS_INF_ADJ⟩ → ‘meet’ | ‘see’
 ⟨VP1_EMBEDDED⟩ → ⟨V_EMBEDDING_PAST⟩ ‘that’ ⟨NAME2⟩ ⟨V_TRANS_PAST_EMBEDDED⟩
 ⟨V_EMBEDDING_PAST⟩ → ‘said’ | ‘insisted’
 ⟨V_TRANS_PAST_EMBEDDED⟩ → ‘met’ | ‘saw’
 ⟨VP2⟩ → ⟨VP2_COMPLEX⟩ | ⟨VP2_SIMPLE⟩
 ⟨VP2_SIMPLE⟩ → ‘hug’ | ‘slap’ | ‘kiss’
 ⟨VP2_COMPLEX⟩ → ⟨VP2_ABOUT⟩ | ⟨VP2_TO⟩ | ⟨VP2_ADJ⟩ | ⟨VP2_EMBEDDED⟩
 ⟨VP2_ABOUT⟩ → ⟨V_ABOUT_FUTURE⟩ ‘to’ ⟨NAME4⟩ ‘about’
 ⟨V_ABOUT_FUTURE⟩ → ‘complain’ | ‘write’
 ⟨VP2_TO⟩ → ⟨V_TO_FUTURE⟩ ⟨NAME4⟩ ‘to’ ⟨V_TRANS_INF_TO_FUTURE⟩
 ⟨V_TO_FUTURE⟩ → ‘encourage’ | ‘beg’
 ⟨V_TRANS_INF_TO_FUTURE⟩ → ‘hug’ | ‘slap’ | ‘kiss’
 ⟨VP2_ADJ⟩ → ‘be’ ⟨ADJ2⟩ ‘to’ ⟨V_TRANS_INF_ADJ2⟩
 ⟨ADJ2⟩ → ‘afraid’ | ‘glad’
 ⟨V_TRANS_INF_ADJ2⟩ → ‘hug’ | ‘slap’ | ‘kiss’
 ⟨VP2_EMBEDDED⟩ → ⟨V_EMBEDDING_FUTURE⟩ ‘that’ ⟨NAME4⟩ ‘will’ ⟨V_TRANS_INF_TO_FUTURE_EMBEDDED⟩
 ⟨V_EMBEDDING_FUTURE⟩ → ‘claim’ | ‘predict’
 ⟨V_TRANS_FUTURE⟩ → ‘hug’ | ‘slap’ | ‘kiss’

B Appendix: Model Failures

Worst 5 four-tuples of sentences per phenomenon (PG, ATB), per model (CHILDES LSTM/Transformer, Wikipedia LSTM/Transformer, GPT-2, GPT-j, GPT-3). Surprisal values are given in parentheses at the relevant position.

B.1 PG – CHILDES LSTM

	<i>+gap</i>	<i>-gap</i>
(9) <i>+filler</i>	I know <u>who</u> Bob’s effort to call is going to bother eventually (14.75)	*I know <u>who</u> Bob’s effort to call is going to bother you (0.64) eventually
<i>-filler</i>	*I know <u>that</u> Bob’s effort to call Mary is going to bother eventually (12.74)	I know <u>that</u> Bob’s effort to call Mary is going to bother you (1.70) eventually
$\Delta_{-filler} - \Delta_{+filler} = -3.06$		

	<i>+gap</i>	<i>-gap</i>
(10) <i>+filler</i>	I know <u>who</u> Bob’s effort to call is going to annoy eventually (15.82)	*I know <u>who</u> Bob’s effort to call is going to annoy you (0.39) eventually
<i>-filler</i>	*I know <u>that</u> Bob’s effort to call Mary is going to annoy eventually (13.65)	I know <u>that</u> Bob’s effort to call Mary is going to annoy you (1.23) eventually
$\Delta_{-filler} - \Delta_{+filler} = -3.01$		

	<i>+gap</i>	<i>-gap</i>
(11) <i>+filler</i>	I know <u>who</u> Bob’s effort to call is going to bother soon (13.28)	*I know <u>who</u> Bob’s effort to call is going to bother you (0.64) soon
<i>-filler</i>	*I know <u>that</u> Bob’s effort to call Mary is going to bother soon (11.53)	I know <u>that</u> Bob’s effort to call Mary is going to bother you (1.70) soon
$\Delta_{-filler} - \Delta_{+filler} = -2.80$		

	<i>+gap</i>	<i>-gap</i>
(12) <i>+filler</i>	I know <u>who</u> John’s dancing with is going to disturb eventually (15.03)	*I know <u>who</u> John’s dancing with is going to disturb us (1.82) eventually
<i>-filler</i>	*I know <u>that</u> John’s dancing with Mary is going to disturb eventually (13.69)	I know <u>that</u> John’s dancing with Mary is going to disturb us (3.27) eventually
$\Delta_{-filler} - \Delta_{+filler} = -2.80$		

	<i>+gap</i>	<i>-gap</i>
(13) <i>+filler</i>	I know <u>who</u> Bob’s failure to meet is going to disturb eventually (13.66)	*I know <u>who</u> Bob’s failure to meet is going to disturb us (1.63) eventually
<i>-filler</i>	*I know <u>that</u> Bob’s failure to meet Mary is going to disturb eventually (12.41)	I know <u>that</u> Bob’s failure to meet Mary is going to disturb us (3.17) eventually
$\Delta_{-filler} - \Delta_{+filler} = -2.79$		

B.2 PG – CHILDES Transformer

	<i>+gap</i>	<i>-gap</i>
(14) <i>+filler</i>	I know <u>who</u> Bob’s attempt to play with is going to bother soon (12.45)	*I know <u>who</u> Bob’s attempt to play with is going to bother us (1.55) soon
<i>-filler</i>	*I know <u>that</u> Bob’s attempt to play with Jennifer is going to bother soon (10.69)	I know <u>that</u> Bob’s attempt to play with Jennifer is going to bother us (3.20) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.41$		

	<i>+gap</i>	<i>-gap</i>
(15) <i>+filler</i>	I know <u>who</u> Bob’s effort to play with is going to bother soon (12.27)	*I know <u>who</u> Bob’s effort to play with is going to bother us (1.02) soon
<i>-filler</i>	*I know <u>that</u> Bob’s effort to play with Jennifer is going to bother soon (10.31)	I know <u>that</u> Bob’s effort to play with Jennifer is going to bother us (2.32) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.27$		

	<i>+gap</i>	<i>-gap</i>
(16) <i>+filler</i>	I know <u>who</u> Bob's attempt to play with is about to bother soon (12.78)	*I know <u>who</u> Bob's attempt to play with is about to bother us (1.63) soon
<i>-filler</i>	*I know <u>that</u> Bob's attempt to play with Jennifer is about to bother soon (10.91)	I know <u>that</u> Bob's attempt to play with Jennifer is about to bother us (2.89) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.13$		

	<i>+gap</i>	<i>-gap</i>
(17) <i>+filler</i>	I know <u>who</u> Bob's attempt to play with is expected to bother soon (13.39)	*I know <u>who</u> Bob's attempt to play with is expected to bother us (2.02) soon
<i>-filler</i>	*I know <u>that</u> Bob's attempt to play with Jennifer is expected to bother soon (11.92)	I know <u>that</u> Bob's attempt to play with Jennifer is expected to bother us (3.66) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.10$		

	<i>+gap</i>	<i>-gap</i>
(18) <i>+filler</i>	I know <u>who</u> Bob's failure to play with is going to bother soon (12.32)	*I know <u>who</u> Bob's failure to play with is going to bother us (1.80) soon
<i>-filler</i>	*I know <u>that</u> Bob's failure to play with Jennifer is going to bother soon (10.63)	I know <u>that</u> Bob's failure to play with Jennifer is going to bother us (3.14) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.03$		

B.3 PG – Wikipedia LSTM

	<i>+gap</i>	<i>-gap</i>
(19) <i>+filler</i>	I know <u>who</u> John's intent to call is going to bother eventually (16.59)	*I know <u>who</u> John's intent to call is going to bother us (4.90) eventually
<i>-filler</i>	*I know <u>that</u> John's intent to call Jennifer is going to bother eventually (13.47)	I know <u>that</u> John's intent to call Jennifer is going to bother us (7.20) eventually
$\Delta_{-filler} - \Delta_{+filler} = -5.42$		

	<i>+gap</i>	<i>-gap</i>
(20)	<i>+filler</i> I know <u>who</u> John's failure to call is going to bother eventually (16.07)	*I know <u>who</u> John's failure to call is going to bother us (4.93) eventually
	<i>-filler</i> *I know <u>that</u> John's failure to call Jennifer is going to bother eventually (13.77)	I know <u>that</u> John's failure to call Jennifer is going to bother us (7.89) eventually
$\Delta_{-filler} - \Delta_{+filler} = -5.26$		

	<i>+gap</i>	<i>-gap</i>
(21)	<i>+filler</i> I know <u>who</u> John's failure to call is going to bother soon (15.47)	*I know <u>who</u> John's failure to call is going to bother us (4.93) soon
	<i>-filler</i> *I know <u>that</u> John's failure to call Jennifer is going to bother soon (13.68)	I know <u>that</u> John's failure to call Jennifer is going to bother us (7.89) soon
$\Delta_{-filler} - \Delta_{+filler} = -4.74$		

	<i>+gap</i>	<i>-gap</i>
(22)	<i>+filler</i> I know <u>who</u> John's attempt to talk to is going to bother eventually (15.99)	*I know <u>who</u> John's attempt to talk to is going to bother us (4.42) eventually
	<i>-filler</i> *I know <u>that</u> John's attempt to talk to Jennifer is going to bother eventually (14.78)	I know <u>that</u> John's attempt to talk to Jennifer is going to bother us (7.65) eventually
$\Delta_{-filler} - \Delta_{+filler} = -4.44$		

	<i>+gap</i>	<i>-gap</i>
(23)	<i>+filler</i> I know <u>who</u> John's intent to talk to is going to bother eventually (16.62)	*I know <u>who</u> John's intent to talk to is going to bother you (5.39) eventually
	<i>-filler</i> *I know <u>that</u> John's intent to talk to Jennifer is going to bother eventually (14.89)	I know <u>that</u> John's intent to talk to Jennifer is going to bother you (8.06) eventually
$\Delta_{-filler} - \Delta_{+filler} = -4.41$		

B.4 PG – Wikipedia Transformer

	<i>+gap</i>	<i>-gap</i>	
(24)	<i>+filler</i>	I know <u>who</u> Bob’s attempt to talk to is about to bother soon (11.09)	*I know <u>who</u> Bob’s attempt to talk to is about to bother you (3.73) soon
	<i>-filler</i>	*I know <u>that</u> Bob’s attempt to talk to Jennifer is about to bother soon (7.94)	I know <u>that</u> Bob’s attempt to talk to Jennifer is about to bother you (10.53) soon
$\Delta_{-filler} - \Delta_{+filler} = -9.95$			

	<i>+gap</i>	<i>-gap</i>	
(25)	<i>+filler</i>	I know <u>who</u> Bob’s attempt to talk to is expected to bother soon (10.01)	*I know <u>who</u> Bob’s attempt to talk to is expected to bother you (3.49) soon
	<i>-filler</i>	*I know <u>that</u> Bob’s attempt to talk to Jennifer is expected to bother soon (7.23)	I know <u>that</u> Bob’s attempt to talk to Jennifer is expected to bother you (10.49) soon
$\Delta_{-filler} - \Delta_{+filler} = -9.78$			

	<i>+gap</i>	<i>-gap</i>	
(26)	<i>+filler</i>	I know <u>who</u> Bob’s attempt to talk to is expected to bother eventually (10.76)	*I know <u>who</u> Bob’s attempt to talk to is expected to bother you (3.49) eventually
	<i>-filler</i>	*I know <u>that</u> Bob’s attempt to talk to Jennifer is expected to bother eventually (8.07)	I know <u>that</u> Bob’s attempt to talk to Jennifer is expected to bother you (10.49) eventually
$\Delta_{-filler} - \Delta_{+filler} = -9.70$			

	<i>+gap</i>	<i>-gap</i>
(27)	<i>+filler</i> I know <u>who</u> John's intent to talk to is expected to bother eventually (10.98)	*I know <u>who</u> John's intent to talk to is expected to bother us (4.03) eventually
	<i>-filler</i> *I know <u>that</u> John's intent to talk to Jennifer is expected to bother eventually (8.32)	I know <u>that</u> John's intent to talk to Jennifer is expected to bother us (11.06) eventually
	$\Delta_{-filler} - \Delta_{+filler} = -9.69$	
	<i>+gap</i>	<i>-gap</i>
(28)	<i>+filler</i> I know <u>who</u> Bob's decision to talk to is about to bother soon (10.66)	*I know <u>who</u> Bob's decision to talk to is about to bother you (3.70) soon
	<i>-filler</i> *I know <u>that</u> Bob's decision to talk to Jennifer is about to bother soon (7.83)	I know <u>that</u> Bob's decision to talk to Jennifer is about to bother you (10.50) soon
	$\Delta_{-filler} - \Delta_{+filler} = -9.63$	

B.5 PG – GPT-2

	<i>+gap</i>	<i>-gap</i>
(29)	<i>+filler</i> I know <u>who</u> Bob's talking to is going to annoy eventually (16.51)	*I know <u>who</u> Bob's talking to is going to annoy you (2.77) eventually
	<i>-filler</i> *I know <u>that</u> Bob's talking to Jennifer is going to annoy eventually (16.17)	I know <u>that</u> Bob's talking to Jennifer is going to annoy you (4.70) eventually
	$\Delta_{-filler} - \Delta_{+filler} = -2.26$	
	<i>+gap</i>	<i>-gap</i>
(30)	<i>+filler</i> I know <u>who</u> Bob's decision to meet is about to bother eventually (18.76)	*I know <u>who</u> Bob's decision to meet is about to bother you (2.17) eventually
	<i>-filler</i> *I know <u>that</u> Bob's decision to meet Jennifer is about to bother eventually (18.42)	I know <u>that</u> Bob's decision to meet Jennifer is about to bother you (4.04) eventually
	$\Delta_{-filler} - \Delta_{+filler} = -2.20$	

	<i>+gap</i>	<i>-gap</i>
(31) <i>+filler</i>	I know <u>who</u> John's decision to call is likely to disturb soon (14.28)	*I know <u>who</u> John's decision to call is likely to disturb you (2.73) soon
<i>-filler</i>	*I know <u>that</u> John's decision to call Mary is likely to disturb soon (15.19)	I know <u>that</u> John's decision to call Mary is likely to disturb you (5.84) soon
$\Delta_{-filler} - \Delta_{+filler} = -2.19$		

	<i>+gap</i>	<i>-gap</i>
(32) <i>+filler</i>	I know <u>who</u> John's talking to is likely to disturb eventually (16.82)	*I know <u>who</u> John's talking to is likely to disturb you (1.81) eventually
<i>-filler</i>	*I know <u>that</u> John's talking to Jennifer is likely to disturb eventually (16.53)	I know <u>that</u> John's talking to Jennifer is likely to disturb you (3.68) eventually
$\Delta_{-filler} - \Delta_{+filler} = -2.16$		

	<i>+gap</i>	<i>-gap</i>
(33) <i>+filler</i>	I know <u>who</u> Bob's decision to call is likely to disturb soon (14.31)	*I know <u>who</u> Bob's decision to call is likely to disturb you (2.70) soon
<i>-filler</i>	*I know <u>that</u> Bob's decision to call Mary is likely to disturb soon (15.37)	I know <u>that</u> Bob's decision to call Mary is likely to disturb you (5.91) soon
$\Delta_{-filler} - \Delta_{+filler} = -2.15$		

B.6 PG – GPT-j

	<i>+gap</i>	<i>-gap</i>
(34) <i>+filler</i>	I know <u>who</u> John's attempt to talk to is about to annoy soon (11.48)	*I know <u>who</u> John's attempt to talk to is about to annoy you (1.62) soon
<i>-filler</i>	*I know <u>that</u> John's attempt to talk to Mary is about to annoy soon (12.86)	I know <u>that</u> John's attempt to talk to Mary is about to annoy you (4.70) soon
$\Delta_{-filler} - \Delta_{+filler} = -1.70$		

	<i>+gap</i>	<i>-gap</i>	
(35)	<i>+filler</i>	I know <u>who</u> John's failure to dance with is about to annoy eventually (13.09)	*I know <u>who</u> John's failure to dance with is about to annoy you (1.53) eventually
	<i>-filler</i>	*I know <u>that</u> John's failure to dance with Mary is about to annoy eventually (13.60)	I know <u>that</u> John's failure to dance with Mary is about to annoy you (3.69) eventually
$\Delta_{-filler} - \Delta_{+filler} = -1.66$			

	<i>+gap</i>	<i>-gap</i>	
(36)	<i>+filler</i>	I know <u>who</u> John's attempt to talk to is about to annoy eventually (12.23)	*I know <u>who</u> John's attempt to talk to is about to annoy you (1.62) eventually
	<i>-filler</i>	*I know <u>that</u> John's attempt to talk to Mary is about to annoy eventually (13.71)	I know <u>that</u> John's attempt to talk to Mary is about to annoy you (4.70) eventually
$\Delta_{-filler} - \Delta_{+filler} = -1.59$			

	<i>+gap</i>	<i>-gap</i>	
(37)	<i>+filler</i>	I know <u>who</u> John's decision to dance with is about to annoy eventually (13.70)	*I know <u>who</u> John's decision to dance with is about to annoy you (1.26) eventually
	<i>-filler</i>	*I know <u>that</u> John's decision to dance with Mary is about to annoy eventually (13.85)	I know <u>that</u> John's decision to dance with Mary is about to annoy you (2.95) eventually
$\Delta_{-filler} - \Delta_{+filler} = -1.54$			

	<i>+gap</i>	<i>-gap</i>	
(38)	<i>+filler</i>	I know <u>who</u> John's decision to dance with is about to annoy soon (12.74)	*I know <u>who</u> John's decision to dance with is about to annoy you (1.26) soon
	<i>-filler</i>	*I know <u>that</u> John's decision to dance with Mary is about to annoy soon (12.92)	I know <u>that</u> John's decision to dance with Mary is about to annoy you (2.95) soon
$\Delta_{-filler} - \Delta_{+filler} = -1.51$			

B.7 PG – GPT-3

	<i>+gap</i>	<i>-gap</i>
(39) <i>+filler</i>	I know <u>who</u> Bob's talking to is likely to annoy soon (14.53)	*I know <u>who</u> Bob's talking to is likely to annoy you (3.39) soon
<i>-filler</i>	*I know <u>that</u> Bob's talking to Mary is likely to annoy soon (13.53)	I know <u>that</u> Bob's talking to Mary is likely to annoy you (7.20) soon
$\Delta_{-filler} - \Delta_{+filler} = -4.81$		

	<i>+gap</i>	<i>-gap</i>
(40) <i>+filler</i>	I know <u>who</u> John's talking to is likely to annoy soon (14.06)	*I know <u>who</u> John's talking to is likely to annoy you (3.55) soon
<i>-filler</i>	*I know <u>that</u> John's talking to Mary is likely to annoy soon (13.59)	I know <u>that</u> John's talking to Mary is likely to annoy you (7.60) soon
$\Delta_{-filler} - \Delta_{+filler} = -4.52$		

	<i>+gap</i>	<i>-gap</i>
(41) <i>+filler</i>	I know <u>who</u> Bob's playing with is likely to annoy soon (13.36)	*I know <u>who</u> Bob's playing with is likely to annoy you (2.75) soon
<i>-filler</i>	*I know <u>that</u> Bob's playing with Mary is likely to annoy soon (13.46)	I know <u>that</u> Bob's playing with Mary is likely to annoy you (6.94) soon
$\Delta_{-filler} - \Delta_{+filler} = -4.09$		

	<i>+gap</i>	<i>-gap</i>
(42) <i>+filler</i>	I know <u>who</u> John's talking to is expected to annoy soon (12.97)	*I know <u>who</u> John's talking to is expected to annoy you (3.05) soon
<i>-filler</i>	*I know <u>that</u> John's talking to Mary is expected to annoy soon (12.61)	I know <u>that</u> John's talking to Mary is expected to annoy you (6.56) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.87$		

	<i>+gap</i>	<i>-gap</i>
(43) <i>+filler</i>	I know <u>who</u> Bob's talking to is likely to annoy eventually (15.21)	*I know <u>who</u> Bob's talking to is likely to annoy you (3.39) eventually
<i>-filler</i>	*I know <u>that</u> Bob's talking to Mary is likely to annoy eventually (15.18)	I know <u>that</u> Bob's talking to Mary is likely to annoy you (7.20) eventually
$\Delta_{-filler} - \Delta_{+filler} = -3.84$		

B.8 ATB – CHILDES LSTM

	<i>+gap</i>	<i>-gap</i>
(44) <i>+filler</i>	I know <u>who</u> John told Mary about lately and is going to encourage Patricia to slap now (9.84)	*I know <u>who</u> John told Mary about lately and is going to encourage Patricia to slap you (1.88) now
<i>-filler</i>	*I know <u>that</u> John told Mary about Bob lately and is going to encourage Patricia to slap now (9.24)	I know <u>that</u> John told Mary about Bob lately and is going to encourage Patricia to slap you (3.35) now
$\Delta_{-filler} - \Delta_{+filler} = -2.08$		

	<i>+gap</i>	<i>-gap</i>
(45) <i>+filler</i>	I know <u>who</u> John insisted that Mary met recently and is going to be afraid to kiss now (8.53)	*I know <u>who</u> John insisted that Mary met recently and is going to be afraid to kiss Kim (10.20) now
<i>-filler</i>	*I know <u>that</u> John insisted that Mary met Bob recently and is going to be afraid to kiss now (8.46)	I know <u>that</u> John insisted that Mary met Bob recently and is going to be afraid to kiss Kim (12.17) now
$\Delta_{-filler} - \Delta_{+filler} = -2.04$		

	<i>+gap</i>	<i>-gap</i>
(46) <i>+filler</i>	I know <u>who</u> John insisted that Mary met recently and is going to be afraid to kiss today (8.56)	*I know <u>who</u> John insisted that Mary met recently and is going to be afraid to kiss Kim (10.20) today
<i>-filler</i>	*I know <u>that</u> John insisted that Mary met Bob recently and is going to be afraid to kiss today (8.53)	I know <u>that</u> John insisted that Mary met Bob recently and is going to be afraid to kiss Kim (12.17) today
$\Delta_{-filler} - \Delta_{+filler} = -2.00$		

	<i>+gap</i>	<i>-gap</i>
(47) <i>+filler</i>	I know <u>who</u> John insisted that Mary met recently and is going to be afraid to slap now (9.34)	*I know <u>who</u> John insisted that Mary met recently and is going to be afraid to slap Kim (9.90) now
<i>-filler</i>	*I know <u>that</u> John insisted that Mary met Bob recently and is going to be afraid to slap now (9.29)	I know <u>that</u> John insisted that Mary met Bob recently and is going to be afraid to slap Kim (11.85) now
$\Delta_{-filler} - \Delta_{+filler} = -2.00$		

	<i>+gap</i>	<i>-gap</i>
(48) <i>+filler</i>	I know <u>who</u> John wanted Mary to invite recently and is going to be afraid to slap now (9.41)	*I know <u>who</u> John wanted Mary to invite recently and is going to be afraid to slap Kim (10.37) now
<i>-filler</i>	*I know <u>that</u> John wanted Mary to invite Bob recently and is going to be afraid to slap now (9.22)	I know <u>that</u> John wanted Mary to invite Bob recently and is going to be afraid to slap Kim (12.16) now
$\Delta_{-filler} - \Delta_{+filler} = -1.98$		

B.9 ATB – CHILDES Transformer

	<i>+gap</i>	<i>-gap</i>	
(49)	<i>+filler</i>	I know <u>who</u> John asked Mary about lately and is going to be afraid to slap now (10.21)	*I know <u>who</u> John asked Mary about lately and is going to be afraid to slap us (3.41) now
	<i>-filler</i>	*I know <u>that</u> John asked Mary about Bob lately and is going to be afraid to slap now (9.42)	I know <u>that</u> John asked Mary about Bob lately and is going to be afraid to slap us (4.14) now
$\Delta_{-filler} - \Delta_{+filler} = -1.51$			

	<i>+gap</i>	<i>-gap</i>	
(50)	<i>+filler</i>	I know <u>who</u> John said that Mary saw recently and is going to slap now (10.72)	*I know <u>who</u> John said that Mary saw recently and is going to slap us (3.14) now
	<i>-filler</i>	*I know <u>that</u> John said that Mary saw Bob recently and is going to slap now (10.18)	I know <u>that</u> John said that Mary saw Bob recently and is going to slap us (4.00) now
$\Delta_{-filler} - \Delta_{+filler} = -1.40$			

	<i>+gap</i>	<i>-gap</i>	
(51)	<i>+filler</i>	I know <u>who</u> John said that Mary saw lately and is going to be afraid to slap now (9.88)	*I know <u>who</u> John said that Mary saw lately and is going to be afraid to slap us (3.24) now
	<i>-filler</i>	*I know <u>that</u> John said that Mary saw Bob lately and is going to be afraid to slap now (9.45)	I know <u>that</u> John said that Mary saw Bob lately and is going to be afraid to slap us (4.21) now
$\Delta_{-filler} - \Delta_{+filler} = -1.40$			

	<i>+gap</i>	<i>-gap</i>
(52) <i>+filler</i>	I know <u>who</u> John asked Mary to call lately and is going to be afraid to slap now (10.08)	*I know <u>who</u> John asked Mary to call lately and is going to be afraid to slap us (3.59) now
<i>-filler</i>	*I know <u>that</u> John asked Mary to call Bob lately and is going to be afraid to slap now (9.83)	I know <u>that</u> John asked Mary to call Bob lately and is going to be afraid to slap us (4.72) now
$\Delta_{-filler} - \Delta_{+filler} = -1.38$		

	<i>+gap</i>	<i>-gap</i>
(53) <i>+filler</i>	I know <u>who</u> John said that Mary saw recently and is going to be afraid to slap now (10.12)	*I know <u>who</u> John said that Mary saw recently and is going to be afraid to slap us (3.78) now
<i>-filler</i>	*I know <u>that</u> John said that Mary saw Bob recently and is going to be afraid to slap now (9.98)	I know <u>that</u> John said that Mary saw Bob recently and is going to be afraid to slap us (5.01) now
$\Delta_{-filler} - \Delta_{+filler} = -1.37$		

B.10 ATB – Wikipedia LSTM

	<i>+gap</i>	<i>-gap</i>
(54) <i>+filler</i>	I know <u>who</u> John asked Mary about recently and is going to kiss soon (14.80)	*I know <u>who</u> John asked Mary about recently and is going to kiss us (8.39) soon
<i>-filler</i>	*I know <u>that</u> John asked Mary about Bob recently and is going to kiss soon (14.40)	I know <u>that</u> John asked Mary about Bob recently and is going to kiss us (11.63) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.64$		

	<i>+gap</i>	<i>-gap</i>
(55) <i>+filler</i>	I know <u>who</u> John met recently and is going to beg Patricia to kiss soon (15.45)	*I know <u>who</u> John met recently and is going to beg Patricia to kiss us (10.27) soon
<i>-filler</i>	*I know <u>that</u> John met Bob recently and is going to beg Patricia to kiss soon (15.46)	I know <u>that</u> John met Bob recently and is going to beg Patricia to kiss us (13.61) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.33$		

	<i>+gap</i>	<i>-gap</i>
(56) <i>+filler</i>	I know <u>who</u> John asked Mary about recently and is going to be glad to kiss soon (14.47)	*I know <u>who</u> John asked Mary about recently and is going to be glad to kiss us (9.19) soon
<i>-filler</i>	*I know <u>that</u> John asked Mary about Bob recently and is going to be glad to kiss soon (14.54)	I know <u>that</u> John asked Mary about Bob recently and is going to be glad to kiss us (12.59) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.33$		

	<i>+gap</i>	<i>-gap</i>
(57) <i>+filler</i>	I know <u>who</u> John was happy to meet recently and is going to kiss soon (13.61)	*I know <u>who</u> John was happy to meet recently and is going to kiss you (7.32) soon
<i>-filler</i>	*I know <u>that</u> John was happy to meet Bob recently and is going to kiss soon (13.36)	I know <u>that</u> John was happy to meet Bob recently and is going to kiss you (10.36) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.29$		

	<i>+gap</i>	<i>-gap</i>	
(58)	<i>+filler</i>	I know <u>who</u> John told Mary about recently and is going to beg Patricia to kiss soon (15.49)	*I know <u>who</u> John told Mary about recently and is going to beg Patricia to kiss us (11.17) soon
	<i>-filler</i>	*I know <u>that</u> John told Mary about Bob recently and is going to beg Patricia to kiss soon (15.26)	I know <u>that</u> John told Mary about Bob recently and is going to beg Patricia to kiss us (14.20) soon
$\Delta_{-filler} - \Delta_{+filler} = -3.26$			

B.11 ATB – Wikipedia Transformer

	<i>+gap</i>	<i>-gap</i>	
(59)	<i>+filler</i>	I know <u>who</u> John saw lately and is going to write to Patricia about now (8.22)	*I know <u>who</u> John saw lately and is going to write to Patricia about us (7.18) now
	<i>-filler</i>	*I know <u>that</u> John saw Bob lately and is going to write to Patricia about now (8.21)	I know <u>that</u> John saw Bob lately and is going to write to Patricia about us (8.38) now
$\Delta_{-filler} - \Delta_{+filler} = -1.21$			

	<i>+gap</i>	<i>-gap</i>	
(60)	<i>+filler</i>	I know <u>who</u> John saw recently and is going to write to Patricia about now (8.01)	*I know <u>who</u> John saw recently and is going to write to Patricia about us (7.28) now
	<i>-filler</i>	*I know <u>that</u> John saw Bob recently and is going to write to Patricia about now (8.11)	I know <u>that</u> John saw Bob recently and is going to write to Patricia about us (8.53) now
$\Delta_{-filler} - \Delta_{+filler} = -1.15$			

	<i>+gap</i>	<i>-gap</i>
(61)	<i>+filler</i>	I know <u>who</u> John met lately and is going to write to Patricia about now (8.26)
	<i>-filler</i>	*I know <u>that</u> John met Bob lately and is going to write to Patricia about now (8.35)
		*I know <u>who</u> John met lately and is going to write to Patricia about us (7.40) now
		I know <u>that</u> John met Bob lately and is going to write to Patricia about us (8.58) now
$\Delta_{-filler} - \Delta_{+filler} = -1.09$		

	<i>+gap</i>	<i>-gap</i>
(62)	<i>+filler</i>	I know <u>who</u> John was happy to see lately and is going to write to Patricia about now (8.46)
	<i>-filler</i>	*I know <u>that</u> John was happy to see Bob lately and is going to write to Patricia about now (8.45)
		*I know <u>who</u> John was happy to see lately and is going to write to Patricia about us (7.52) now
		I know <u>that</u> John was happy to see Bob lately and is going to write to Patricia about us (8.54) now
$\Delta_{-filler} - \Delta_{+filler} = -1.04$		

	<i>+gap</i>	<i>-gap</i>
(63)	<i>+filler</i>	I know <u>who</u> John was happy to meet lately and is going to write to Patricia about now (8.51)
	<i>-filler</i>	*I know <u>that</u> John was happy to meet Bob lately and is going to write to Patricia about now (8.57)
		*I know <u>who</u> John was happy to meet lately and is going to write to Patricia about us (7.68) now
		I know <u>that</u> John was happy to meet Bob lately and is going to write to Patricia about us (8.72) now
$\Delta_{-filler} - \Delta_{+filler} = -0.98$		

B.12 ATB – GPT-2

	<i>+gap</i>	<i>-gap</i>	
(64)	<i>+filler</i>	I know <u>who</u> John said that Mary saw lately and is going to be glad to kiss soon (14.91)	*I know <u>who</u> John said that Mary saw lately and is going to be glad to kiss you (4.06) soon
	<i>-filler</i>	*I know <u>that</u> John said that Mary saw Bob lately and is going to be glad to kiss soon (16.39)	I know <u>that</u> John said that Mary saw Bob lately and is going to be glad to kiss you (7.08) soon
$\Delta_{-filler} - \Delta_{+filler} = -1.53$			

	<i>+gap</i>	<i>-gap</i>	
(65)	<i>+filler</i>	I know <u>who</u> John said that Mary met lately and is going to be glad to kiss soon (14.46)	*I know <u>who</u> John said that Mary met lately and is going to be glad to kiss you (4.11) soon
	<i>-filler</i>	*I know <u>that</u> John said that Mary met Bob lately and is going to be glad to kiss soon (15.85)	I know <u>that</u> John said that Mary met Bob lately and is going to be glad to kiss you (6.93) soon
$\Delta_{-filler} - \Delta_{+filler} = -1.43$			

	<i>+gap</i>	<i>-gap</i>	
(66)	<i>+filler</i>	I know <u>who</u> John said that Mary met lately and is going to be glad to kiss now (12.15)	*I know <u>who</u> John said that Mary met lately and is going to be glad to kiss you (4.11) now
	<i>-filler</i>	*I know <u>that</u> John said that Mary met Bob lately and is going to be glad to kiss now (13.67)	I know <u>that</u> John said that Mary met Bob lately and is going to be glad to kiss you (6.93) now
$\Delta_{-filler} - \Delta_{+filler} = -1.29$			

	<i>+gap</i>	<i>-gap</i>	
(67)	<i>+filler</i>	I know <u>who</u> John said that Mary met lately and is going to kiss soon (13.26)	*I know <u>who</u> John said that Mary met lately and is going to kiss you (5.31) soon
	<i>-filler</i>	*I know <u>that</u> John said that Mary met Bob lately and is going to kiss soon (15.08)	I know <u>that</u> John said that Mary met Bob lately and is going to kiss you (8.30) soon
$\Delta_{-filler} - \Delta_{+filler} = -1.18$			

	<i>+gap</i>	<i>-gap</i>	
(68)	<i>+filler</i>	I know <u>who</u> John was eager to see lately and is going to predict that Patricia will kiss soon (11.50)	*I know <u>who</u> John was eager to see lately and is going to predict that Patricia will kiss Kim (10.84) soon
	<i>-filler</i>	*I know <u>that</u> John was eager to see Bob lately and is going to predict that Patricia will kiss soon (11.45)	I know <u>that</u> John was eager to see Bob lately and is going to predict that Patricia will kiss Kim (11.95) soon
$\Delta_{-filler} - \Delta_{+filler} = -1.16$			

B.13 ATB – GPT-j

	<i>+gap</i>	<i>-gap</i>	
(69)	<i>+filler</i>	I know <u>who</u> John told Mary about lately and is going to hug now (8.12)	*I know <u>who</u> John told Mary about lately and is going to hug us (6.59) now
	<i>-filler</i>	*I know <u>that</u> John told Mary about Bob lately and is going to hug now (8.88)	I know <u>that</u> John told Mary about Bob lately and is going to hug us (9.42) now
$\Delta_{-filler} - \Delta_{+filler} = -2.07$			

	<i>+gap</i>	<i>-gap</i>
(70) <i>+filler</i>	I know <u>who</u> John told Mary about lately and is going to hug today (8.69)	*I know <u>who</u> John told Mary about lately and is going to hug us (6.59) today
<i>-filler</i>	*I know <u>that</u> John told Mary about Bob lately and is going to hug today (9.54)	I know <u>that</u> John told Mary about Bob lately and is going to hug us (9.42) today
$\Delta_{-filler} - \Delta_{+filler} = -1.98$		

	<i>+gap</i>	<i>-gap</i>
(71) <i>+filler</i>	I know <u>who</u> John said that Mary met recently and is going to be afraid to hug now (7.54)	*I know <u>who</u> John said that Mary met recently and is going to be afraid to hug us (4.59) now
<i>-filler</i>	*I know <u>that</u> John said that Mary met Bob recently and is going to be afraid to hug now (9.06)	I know <u>that</u> John said that Mary met Bob recently and is going to be afraid to hug us (8.07) now
$\Delta_{-filler} - \Delta_{+filler} = -1.96$		

	<i>+gap</i>	<i>-gap</i>
(72) <i>+filler</i>	I know <u>who</u> John said that Mary met recently and is going to be afraid to slap now (10.08)	*I know <u>who</u> John said that Mary met recently and is going to be afraid to slap us (4.22) now
<i>-filler</i>	*I know <u>that</u> John said that Mary met Bob recently and is going to be afraid to slap now (10.79)	I know <u>that</u> John said that Mary met Bob recently and is going to be afraid to slap us (6.81) now
$\Delta_{-filler} - \Delta_{+filler} = -1.88$		

	<i>+gap</i>	<i>-gap</i>
(73) <i>+filler</i>	I know <u>who</u> John told Mary about recently and is going to be afraid to hug today (10.32)	*I know <u>who</u> John told Mary about recently and is going to be afraid to hug us (6.16) today
<i>-filler</i>	*I know <u>that</u> John told Mary about Bob recently and is going to be afraid to hug today (11.78)	I know <u>that</u> John told Mary about Bob recently and is going to be afraid to hug us (9.46) today
$\Delta_{-filler} - \Delta_{+filler} = -1.84$		

B.14 ATB – GPT-3

	<i>+gap</i>	<i>-gap</i>
(74) <i>+filler</i>	I know <u>who</u> John told Mary about lately and is going to be afraid to hug now (8.21)	*I know <u>who</u> John told Mary about lately and is going to be afraid to hug you (5.06) now
<i>-filler</i>	*I know <u>that</u> John told Mary about Bob lately and is going to be afraid to hug now (8.39)	I know <u>that</u> John told Mary about Bob lately and is going to be afraid to hug you (8.09) now
$\Delta_{-filler} - \Delta_{+filler} = -2.85$		

	<i>+gap</i>	<i>-gap</i>
(75) <i>+filler</i>	I know <u>who</u> John told Mary about lately and is going to slap now (9.43)	*I know <u>who</u> John told Mary about lately and is going to slap you (4.70) now
<i>-filler</i>	*I know <u>that</u> John told Mary about Bob lately and is going to slap now (10.25)	I know <u>that</u> John told Mary about Bob lately and is going to slap you (8.27) now
$\Delta_{-filler} - \Delta_{+filler} = -2.75$		

	<i>+gap</i>	<i>-gap</i>
(76)	<i>+filler</i> I know <u>who</u> John told Mary about lately and is going to slap soon (7.22)	*I know <u>who</u> John told Mary about lately and is going to slap you (4.70) soon
	<i>-filler</i> *I know <u>that</u> John told Mary about Bob lately and is going to slap soon (8.33)	I know <u>that</u> John told Mary about Bob lately and is going to slap you (8.27) soon
$\Delta_{-filler} - \Delta_{+filler} = -2.47$		

	<i>+gap</i>	<i>-gap</i>
(77)	<i>+filler</i> I know <u>who</u> John told Mary about lately and is going to be glad to slap soon (9.32)	*I know <u>who</u> John told Mary about lately and is going to be glad to slap you (3.35) soon
	<i>-filler</i> *I know <u>that</u> John told Mary about Bob lately and is going to be glad to slap soon (9.52)	I know <u>that</u> John told Mary about Bob lately and is going to be glad to slap you (5.82) soon
$\Delta_{-filler} - \Delta_{+filler} = -2.27$		

	<i>+gap</i>	<i>-gap</i>
(78)	<i>+filler</i> I know <u>who</u> John told Mary about lately and is going to be glad to slap today (10.41)	*I know <u>who</u> John told Mary about lately and is going to be glad to slap you (3.35) today
	<i>-filler</i> *I know <u>that</u> John told Mary about Bob lately and is going to be glad to slap today (10.64)	I know <u>that</u> John told Mary about Bob lately and is going to be glad to slap you (5.82) today
$\Delta_{-filler} - \Delta_{+filler} = -2.24$		

C Appendix: Context-Free Grammars for Retraining Task (Section 5)

C.1 PG – retraining corpus

$\langle S \rangle \rightarrow \langle S_FG \rangle$
 $\langle S_FG \rangle \rightarrow \langle PREAMBLE \rangle \langle F \rangle \langle G \rangle$
 $\langle S_XG \rangle \rightarrow \langle UNGRAMMATICAL \rangle \langle PREAMBLE \rangle \langle XF \rangle \langle G \rangle$
 $\langle S_FX \rangle \rightarrow \langle UNGRAMMATICAL \rangle \langle PREAMBLE \rangle \langle F \rangle \langle XG \rangle$
 $\langle S_XX \rangle \rightarrow \langle PREAMBLE \rangle \langle XF \rangle \langle XG \rangle$
 $\langle GEN \rangle \rightarrow \text{'s'}$
 $\langle OBJ \rangle \rightarrow \text{'you' | 'us' | 'Kim'}$
 $\langle NAME1 \rangle \rightarrow \text{'Bob' | 'John'}$
 $\langle NAME2 \rangle \rightarrow \text{'Mary' | 'Jennifer'}$
 $\langle NAME3 \rangle \rightarrow \text{'James' | 'Michael'}$
 $\langle NAME4 \rangle \rightarrow \text{'Patricia' | 'Linda'}$
 $\langle PREAMBLE \rangle \rightarrow \text{'I know'}$
 $\langle F \rangle \rightarrow \text{'who' } \langle NAME1 \rangle \langle GEN \rangle \langle SUBJ \rangle$
 $\langle XF \rangle \rightarrow \text{'that' } \langle NAME1 \rangle \langle GEN \rangle \langle SUBJ \rangle \langle NAME2 \rangle$
 $\langle G \rangle \rightarrow \langle G_PAST \rangle | \langle G_FUTURE \rangle$
 $\langle XG \rangle \rightarrow \langle XG_PAST \rangle | \langle XG_FUTURE \rangle$
 $\langle G_PAST \rangle \rightarrow \langle VP_PAST \rangle \langle ADV_PAST \rangle$
 $\langle G_FUTURE \rangle \rightarrow \langle VP_FUTURE \rangle \langle ADV_FUTURE \rangle$
 $\langle XG_PAST \rangle \rightarrow \langle VP_PAST \rangle \langle XG_OBJ \rangle \langle ADV_PAST \rangle$
 $\langle XG_FUTURE \rangle \rightarrow \langle VP_FUTURE \rangle \langle XG_OBJ \rangle \langle ADV_FUTURE \rangle$
 $\langle VP_PAST \rangle \rightarrow \text{'upset' | 'distracted' | 'worried' | 'annoyed' | 'amused' | 'delighted'}$
 $\langle VP_FUTURE \rangle \rightarrow \text{'will' } \langle V_FUTURE \rangle$
 $\langle V_FUTURE \rangle \rightarrow \text{'upset' | 'distract' | 'worry' | 'annoy' | 'amuse' | 'delight'}$
 $\langle XG_OBJ \rangle \rightarrow \langle NAME4 \rangle | \langle OBJ \rangle$
 $\langle SUBJ \rangle \rightarrow \text{'attitude towards' | 'friendship with' | 'praising of' | 'fight with' | 'kissing with' | 'asking about'}$
 $\langle ADV_PAST \rangle \rightarrow \text{'yesterday' | 'recently' | 'often' | 'constantly' | 'today' | 'lately' | 'earlier'}$
 $\langle ADV_FUTURE \rangle \rightarrow \text{'today' | 'soon' | 'tomorrow' | 'now' | 'quickly'}$

C.2 ATB – retraining corpus

$\langle S \rangle \rightarrow \langle S_FG \rangle$
 $\langle S_FG \rangle \rightarrow \langle PREAMBLE \rangle \langle F \rangle \langle G \rangle$
 $\langle S_XG \rangle \rightarrow \langle UNGRAMMATICAL \rangle \langle PREAMBLE \rangle \langle XF \rangle \langle G \rangle$
 $\langle S_FX \rangle \rightarrow \langle UNGRAMMATICAL \rangle \langle PREAMBLE \rangle \langle F \rangle \langle XG \rangle$
 $\langle S_XX \rangle \rightarrow \langle PREAMBLE \rangle \langle XF \rangle \langle XG \rangle$
 $\langle GEN \rangle \rightarrow \text{'s'}$
 $\langle OBJ \rangle \rightarrow \text{'you' | 'us' | 'Kim'}$
 $\langle NAME1 \rangle \rightarrow \text{'John'}$
 $\langle NAME2 \rangle \rightarrow \text{'Mary'}$
 $\langle NAME3 \rangle \rightarrow \text{'Bob'}$
 $\langle NAME4 \rangle \rightarrow \text{'Patricia'}$
 $\langle PREAMBLE \rangle \rightarrow \text{'I know'}$
 $\langle F \rangle \rightarrow \text{'who' } \langle SUBJ_F \rangle$
 $\langle XF \rangle \rightarrow \text{'that' } \langle SUBJ_XF \rangle$
 $\langle CONN \rangle \rightarrow \text{'and'}$
 $\langle SUBJ_F \rangle \rightarrow \langle ONE_SUBJ_F \rangle \mid \langle TWO_SUBJ_F \rangle$
 $\langle SUBJ_XF \rangle \rightarrow \langle ONE_SUBJ_XF \rangle \mid \langle TWO_SUBJ_XF \rangle$
 $\langle ONE_SUBJ_F \rangle \rightarrow \langle NAME1 \rangle \langle V1 \rangle \langle ADV_PAST_1 \rangle \langle CONN \rangle$
 $\langle TWO_SUBJ_F \rangle \rightarrow \langle NAME1 \rangle \langle V1 \rangle \langle ADV_PAST_1 \rangle \langle CONN \rangle \langle NAME3 \rangle$
 $\langle ONE_SUBJ_XF \rangle \rightarrow \langle NAME1 \rangle \langle V1 \rangle \langle NAME2 \rangle \langle ADV_PAST_1 \rangle \langle CONN \rangle$
 $\langle TWO_SUBJ_XF \rangle \rightarrow \langle NAME1 \rangle \langle V1 \rangle \langle NAME2 \rangle \langle ADV_PAST_1 \rangle \langle CONN \rangle \langle NAME3 \rangle$
 $\langle G \rangle \rightarrow \langle G_PAST \rangle \mid \langle G_FUTURE \rangle$
 $\langle XG \rangle \rightarrow \langle XG_PAST \rangle \mid \langle XG_FUTURE \rangle$
 $\langle XG_PAST \rangle \rightarrow \langle VP2_PAST \rangle \langle XG_OBJ \rangle \langle ADV_PAST_2 \rangle$
 $\langle XG_FUTURE \rangle \rightarrow \langle VP2_FUTURE \rangle \langle XG_OBJ \rangle \langle ADV_FUTURE \rangle$
 $\langle G_PAST \rangle \rightarrow \langle VP2_PAST \rangle \langle ADV_PAST_2 \rangle$
 $\langle G_FUTURE \rangle \rightarrow \langle VP2_FUTURE \rangle \langle ADV_FUTURE \rangle$
 $\langle XG_OBJ \rangle \rightarrow \langle NAME4 \rangle \mid \langle OBJ \rangle$
 $\langle V1 \rangle \rightarrow \text{'saw' | 'hugged' | 'helped' | 'met' | 'pushed' | 'praised' | 'chased' | 'hired' | 'invited' | 'promoted'}$
 $\langle VP2 \rangle \rightarrow \langle VP2_PAST \rangle \mid \langle VP2_FUTURE \rangle$
 $\langle VP2_PAST \rangle \rightarrow \langle V2_PAST \rangle$
 $\langle V2_PAST \rangle \rightarrow \text{'kissed' | 'slapped' | 'insulted' | 'annoyed' | 'hurt' | 'mocked' | 'teased' | 'supported' | 'm'}$
 $\langle VP2_FUTURE \rangle \rightarrow \text{'will' } \langle V2_FUTURE \rangle$
 $\langle V2_FUTURE \rangle \rightarrow \text{'kiss' | 'slap' | 'insult' | 'annoy' | 'hurt' | 'mock' | 'tease' | 'support' | 'marry'}$
 $\langle ADV_PAST_1 \rangle \rightarrow \text{'yesterday' | 'recently' | 'often' | 'constantly'}$
 $\langle ADV_PAST_2 \rangle \rightarrow \text{'today' | 'lately' | 'earlier' | 'regularly' | 'repeatedly'}$

$\langle ADV_FUTURE \rangle \rightarrow$ 'today' | 'soon' | 'tomorrow' | 'now' | 'quickly'

Résumé français

Ce résumé est une version informelle, le lecteur est invité à se référer aux documents de référence disponibles dans le cœur de la thèse, pour un résumé comme pour les détails des résultats.

L'objectif de cette thèse est de déterminer comment les réseaux de neurones artificiels (RNA) diffèrent des humains en termes de capacités de généralisation, et comment alors réduire les différences potentielles. Les RNA ont connu une croissance fulgurante ces dernières années, à une croissance à la fois en termes de leur succès et de leur taille. L'hypothèse principale de cette thèse est qu'en imposant des contraintes de simplicité rationnelles, de plus petits RNA pourraient être en mesure d'effectuer de meilleures généralisations.

Cette thèse présente une nouvelle proposition, suivie de trois études. Dans le chapitre 1, nous proposons un nouveau type de réseaux neuronaux récurrents (RNR), dans lesquels les objectifs d'apprentissage standards sont remplacés/complétés par un objectif de minimisation de la longueur de description minimale (Minimum Description Length – MDL) du réseau lui-même. En appliquant ce principe, on obtient intentionnellement des RNN beaucoup plus petits qui peuvent apprendre, en toute généralité, certains langages formels qui étaient hors de portée des ANNs standards.

Les chapitres 2 et 3 suggèrent donc qu'un changement d'objectif d'apprentissage pourrait être nécessaire pour atteindre une généralisation de niveau humain dans les RNAs. Dans le chapitre 2, nous proposons un moyen systématique d'inspecter

les capacités de généralisation des modèles artificiels à l'aide de langages formels. Nous publions plusieurs ensembles de données d'étalonnage qui unifient et normalisent les résultats précédents présents dans la littérature de façon dispersés. Nous montrons que les performances des RNA existants, même ceux équipés d'architectures plus expressives, sont inférieures à celles du modèle MDL du chapitre 1.

Dans le chapitre 3, nous construisons manuellement un réseau qui capture parfaitement le langage $a^n b^n$, et nous montrons qu'il ne se situe pas aux optima des objectifs standards. En revanche, ce réseau parfait en terme de comportement, est bel et bien un optimum pour l'objectif MDL que nous proposons.

Dans le chapitre 4, nous mettons de côté la question de l'objectif d'apprentissage et examinons les connaissances linguistiques acquises par les grands modèles de langage (LLM). Pour ce faire, nous considérons l'argument de la pauvreté du stimulus (APS) - l'argument longtemps débattu selon laquelle l'apprentissage du langage doit reposer sur des capacités innées chez les humains, sans quoi l'input linguistique à la disposition d'un enfant ne lui permettrait pas de faire les généralisations qu'il fait. Nous montrons que, contrairement à de précédents résultats, les LLM ont du mal à acquérir une connaissance satisfaisante de phénomènes syntaxiques pour lesquels les humains ont des jugements clairs ('parasitic gap' et 'across-the-board movement').

Étant donné que les LLM modernes sont entraînés à partir de quantités massives de données, qui dépassent de plusieurs ordres de grandeur l'expérience

linguistique des enfants, nous concluons que ces échecs soutiennent l'affirmation selon laquelle les humains sont dotés de moyens spécifiques qui rendent possible l'acquisition de tels phénomènes à partir de beaucoup moins d'inputs, c'est-à-dire qu'ils soutiennent l'APS. En d'autres termes, les êtres humains ont des biais d'apprentissage, et ce sont ces biais qui leur permettent de converger vers des langues communes à partir d'un input similaire. En réalité, tous les systèmes d'apprentissage ont de tels biais, des priors dit-on dans la terminologie bayésienne, mais leurs biais sont différents.

Chapitre 1

Un système d'apprentissage efficace fait des généralisations appropriées. Par exemple, après avoir vu la séquence 1,0,1,0,1, nous pouvons soupçonner que l'élément suivant sera 0. Si nous voyons alors 0, nous pourrions être encore plus sûrs que l'élément suivant sera 1. Les réseaux de neurones artificiels ont montré des résultats impressionnants dans un large éventail de domaines, notamment les données linguistiques, la vision et bien d'autres. Ils excellent dans la généralisation lorsque de grands corpus de formation et de grandes ressources informatiques sont disponibles, mais ils sont confrontés à de sérieux défis qui deviennent particulièrement évidents lorsqu'ils généralisent à partir de petites séquences d'entrée comme celle présentée ci-dessus.

Premièrement, les réseaux de neurones artificiels ont tendance à suivre de trop près leurs données d'entraînement. Pour éviter cela, ils ont besoin de mesures

extérieures pour contrôler leur propre tendance à la mémorisation (telles que la régularisation) ainsi que de très grands corpus d'entraînement. Pour cela, il existe des techniques dites de régularisation, mais elles échouent dans de nombreux cas, comme nous le montrons ci-dessous.

Deuxièmement, même lorsque ces méthodes de régularisation réussissent, elles ont tendance à produire des résultats non catégoriques. Autrement dit, les réseaux de neurones sont potentiellement capables d'assigner des probabilités très élevées à une bonne réponse, mais jamais 100

Troisièmement, ces réseaux sont souvent très grands, et il est généralement très difficile d'inspecter un réseau donné et de déterminer ce qu'il sait réellement (bien que voir Lakretz et al. 2019 pour une récente tentative réussie d'interroger ces connaissances dans un contexte linguistique).

Certains des défis ci-dessus découlent du recours aux approches connexionnistes courantes sur l'algorithme de "Backpropagation" comme méthode de formation, qui maintient l'architecture neuronale elle-même constante tout au long de la recherche. L'architecture choisie doit donc être suffisamment grande pour répondre à la tâche donnée, et il est naturel de dépasser les limites en termes de taille. De plus, il nécessite que les opérations mises en jeu soient différentiables. Cette contrainte rend inapplicable la méthode pour la recherche de certains modèles catégoriels, voire même pour la possibilité de les exprimer.

Dans cet article, nous proposons d'étudier une méthode d'entraînement qui diffère des approches courantes au sens où une partie de son objectif est d'optimiser

des propriétés structurelles du réseau, la “longueur minimale de description” (Minimum Description Length, MDL; Rissanen:1978).

Cela revient à minimiser l’erreur comme d’habitude, en même temps que de minimiser la taille du réseau lui-même (une pression similaire à un biais bayésien de taille, ou de parcimonie). En conséquence, la fonction objectif offre un guide pour déterminer la taille du réseau (un guide que la minimisation des erreurs à elle seule ne fournit pas), ce qui signifie que l’architecture elle-même peut évoluer au cours de l’apprentissage et peut potentiellement diminuer en taille.

Un effet secondaire potentiel est que l’optimisation ne peut pas être réalisée uniquement par Backpropagation. Nous utilisons ici un algorithme génétique pour effectuer une recherche dans le très vaste espace d’hypothèses de réseaux de neurones de différentes tailles.

Nous constatons que les réseaux optimisés pour leur MDL atteignent des généralisations adéquates à partir de très petits corpus et évitent le “overfit”. Les réseaux optimisés pour MDL sont tous petits et transparents ; nous fournissons des preuves d’exactitude qui se résument à des ensembles de tests infinis et exhaustifs. Ces réseaux peuvent également fournir des résultats déterministes lorsque cela est pertinent (exprimant une confiance pure de 100

Nos résultats montrent qu’un modèle optimisé pour MDL arrive à des réseaux qui sont proches de la véritable distribution avec de petits corpus d’entraînement, pour des tâches qui sont considérées classiquement difficiles. Dans plusieurs cas, les réseaux obtiennent des scores parfaits. Au-delà de l’évaluation habituelle

en termes de performances sur des ensembles de tests, les réseaux se prêtent à une inspection directe et montrent un énoncé explicite du modèle qui a généré le corpus.

Chapitre 2

La mesure dans laquelle les réseaux de neurones artificiels se généralisent au-delà de leurs données de formation reste une question de recherche ouverte.

Dans ce travail, nous abordons cette question du point de vue de l'induction grammaticale, c'est-à-dire l'apprentissage d'une grammaire formelle à partir d'un échantillon fini (souvent petit) d'un langage (généralement infini) de cette grammaire. Pour réussir cette tâche, un modèle d'apprentissage doit trouver un équilibre entre l'ajustement des données d'entraînement et la généralisation à un ensemble potentiellement infini de chaînes invisibles.

Les humains testés sur de telles tâches montrent une généralisation systématique à partir de petits ensembles d'exemples (Fitch et Hauser 2004, Malassis et al. 2020).

Bien qu'il ait été démontré qu'une gamme d'architectures de réseaux de neurones artificiels atteignent des approximations pour les langages formels, la qualité de cette approximation reste une question ouverte, comme nous le montrons ci-dessous.

Ici, nous nous appuyons sur des études précédentes de généralisation de réseaux de neurones artificiels pour l'induction grammaticale et introduisons une

manière unifiée et générale d'évaluer cette capacité, pour une paire donnée d'un modèle d'apprentissage et d'un corpus tiré d'un langage formel.

Nos contributions principales sont :

1. Un benchmark pour l'apprentissage formel des langues. Le benchmark s'appuie sur une méthode de quantification de la généralisation de réseaux de neurones artificiels pour les langages formels, y compris les langages probabilistes.

La méthode attribue un score d'indice représentant les performances de généralisation d'un modèle en relation inverse avec la taille des données d'entraînement. Nous présentons la méthode et fournissons des ensembles de données concrets pour les langages formels les plus couramment étudiés.

2. Une évaluation des architectures sélectionnées. Nous testons des réseaux de neurones récurrents (RNN) de l'architecture Long-Short Term Memory (LSTM ; Hochreiter 1997) ; RNN à mémoire augmentée (MARNN ; Suzgun et al. 2019); et une variante RNN qui remplace le régime d'entraînement traditionnel basé sur le gradient par un objectif qui optimise la longueur minimale de description du réseau (MDLRNN ; Lan et al. 2022).

Nous constatons qu'équiper les réseaux de neurones artificiels de dispositifs de mémoire tels que des stacks différentiables facilite la généralisation, mais la généralisation reste partielle et lente. En revanche, l'entraînement guidé par MDL conduit dans certains des cas que nous avons examinés à une généralisation potentiellement parfaite, en utilisant beaucoup moins de données. Dans d'autres cas, l'entraînement guidé par MDL n'a pas abouti à une généralisation réussie,

potentiellement car la procédure d'optimisation que nous avons utilisée pour la recherche d'architecture n'a pas réussi à trouver l'optimum global sous la fonction objectif MDL.

Nous avons fourni un indice simple indiquant dans quelle mesure un modèle effectue des généralisations : combien peut-il apprendre à partir de peu de données. Nous avons illustré l'utilité de cet indice dans une comparaison de plusieurs modèles actuels apprenant différents langages formels. En plus de montrer quels modèles existants généralisent mieux que d'autres, le benchmark met également en évidence quels aspects des réseaux de neurones artificiels fonctionnent bien pour l'induction grammaticale, et ce qui manque encore.

Parmi les langages appris avec une parfaite précision ($a^n b^n$, $a^n b^m c^{n+m}$, Dyck-1), les MDLRNN ont généralisé le mieux. Ils ont toutefois échoué quand même dans d'autres ($a^n b^n c^n$, $a^n b^n c^n d^n$, et Dyck-2). Des travaux précédents avaient montré que la procédure de recherche de ce modèle, un algorithme génétique, ne parvient dans certains cas pas à trouver des réseaux ayant de meilleurs scores MDL (Lan et al. 2022). Nous interprétons cela comme indice que la procédure d'optimisation limite le modèle et l'empêche de tirer pleinement parti de l'objectif MDL. L'avantage de l'objectif MDL est néanmoins évident dans les performances de généralisation pour plusieurs langages.

Les MARNN bénéficient clairement de leurs dispositifs de mémoire et atteignent de bons scores de généralisation, mais les tests de précision parfaite ($\epsilon=0$) révèlent que leurs résultats d'apprentissage sont pour la plupart ap-

proximatifs et qu'ils ne parviennent pas à maintenir une précision parfaite pendant de longues périodes au-delà de leurs données d'entraînement.

Cela pourrait être le résultat d'une fonction d'objectif inadéquate (cross-entropy), de limitations de l'algorithme de recherche (Backpropagation/gradient descent), ou des deux. Nous ne disposons pas actuellement de résultats permettant de trancher cette question, mais des résultats récents pour d'autres architectures (El-Naggar et al. 2023) suggèrent que le problème réside au moins en partie dans la fonction objectif.

Chapitre 3

L'exploration des capacités des réseaux de neurones artificiels (ANN) dans le domaine de l'apprentissage des langages formels a progressé dans deux voies complémentaires : théorique et empirique. Les travaux théoriques tentent de délimiter les types de langages et de phénomènes qui peuvent être exprimés par les réseaux de neurones artificiels, et les travaux empiriques impliquent de former des réseaux à de telles tâches et d'inspecter leurs performances.

Un fait souvent négligé est que ces chemins n'ont toujours pas convergé : tandis que les travaux théoriques continuent de fournir des résultats encourageants concernant l'expressivité de différentes architectures, les travaux empiriques continuent d'aboutir à des solutions sous-optimales qui sont en deçà des solutions théoriquement correctes.

Par exemple, pour les langages formels tels que $a^n b^n$ ou Dyck-1, entre autres,

nous n'avons conscience d'aucun réseau entraîné par gradient descent qui se soit révélé performant sur des séquences significativement plus longues que les séquences observées lors de l'entraînement. En revanche, les défaillances aux faibles longueurs sont omniprésentes (Joulin et Mikolov 2015, Weiss et al. 2018, Suzgun et al. 2019, Bhattamishra et al. 2020, El-Naggar et al. 2022, entre autres ; voir Lan et al. 2023 pour un aperçu). Cela contraste avec les modèles symboliques, pour lesquels la précision et la stabilité de la solution sur toutes les longueurs sont satisfaites de manière triviale.

La raison pour laquelle cela serait le cas est souvent soit laissée inexplicée, soit considérée comme une lacune de la méthode d'optimisation (le plus souvent, gradient descent utilisant la Backpropagation). Dans ce travail, nous proposons l'idée que ces échecs ne sont pas dus à des problèmes techniques qui pourraient être surmontés, par exemple, en utilisant une recherche plus exhaustive d'hyperparamètres d'optimisation. Ils sont plutôt dus aux caractéristiques inhérentes des objectifs actuellement utilisés pour de telles tâches: l'objectif pourrait être parfaitement satisfait, sans que la solution ne soit celle recherchée.

Nos principales contributions sont :

1. Nous présentons un réseau Long-Short Term Memory (LSTM ; Hochreiter et Schmidhuber 1997) optimal, construit manuellement, qui accepte le langage formel $a^n b^n$, suivant une recette générale donnée par Weiss et al. (2018). Nous montrons que ce réseau ne pourrait pas être trouvé en utilisant des objectifs standards, puisqu'il ne se situe pas aux points optimaux de ces objectifs — même

en utilisant des termes de régularisation qui, selon l'opinion commune, devraient aboutir à des bonnes solutions générales.

2. Nous montrons qu'en remplaçant ces objectifs et termes de régularisation par un objectif de minimiser la longueur minimale de description du réseau (MDL, Rissanen 1978), accompagné d'un schéma de codage intuitif, le réseau optimal devient un optimum de l'objectif.

Nous nous appuyons principalement sur trois travaux récents, que nous prolongeons de la manière suivante. Premièrement, les travaux actuels sont similaires à ceux d'El-Naggar et al. 2023, qui a inspecté le rôle de la fonction objectif dans l'apprentissage des langages formels. Cette œuvre a montré que pour un simple réseau de neurones récurrent (RNN), qui utilise une seule couche ReLU, la solution de comptage optimale ne correspond pas aux optima des fonctions de perte communes (cross-entropy et MSE). Cela a été fait en fournissant les conditions nécessaires et suffisantes pour la mise en œuvre du comptage dans un ReLU-RNN.

Nous étendons ce travail des manières suivantes. Tout d'abord, nous passons au RNN LSTM le plus couramment utilisé. Cette architecture étant plus complexe, il est également plus difficile de trouver des conditions suffisantes et nécessaires pour le comptage, comme le font El-Naggar et al. (2023). Cela laisse nos résultats essentiellement empiriques, par rapport à leur résultat analytique. Cependant, nous allons au-delà de ce travail en proposant également un objectif alternatif (MDL), pour lequel le réseau optimal devient un optimum.

Deuxièmement, afin de localiser un tel optimum de l'objectif, nous construisons un LSTM optimal qui accepte un langage formel spécifique. Pour cela, nous nous appuyons sur Weiss et al. (2018), qui ont montré qu'un LSTM peut théoriquement mettre en œuvre un comptage en utilisant des configurations spécifiques, de sorte que le vecteur de mémoire contienne un compteur qui peut être incrémenté et décrémenté en fonction des entrées présentées au réseau. Ici, nous implémentons leur recette générale pour construire un LSTM optimal qui accepte le langage $a^n b^n$. Nous nous concentrons sur un seul langage afin de simplifier la présentation. La méthode peut être facilement étendue à plusieurs langages.

Troisièmement, et le plus proche des travaux actuels, Lan et al. (2022) ont appliqué le principe MDL aux RNN pour l'apprentissage de langages formels. Les réseaux résultants se sont révélés corrects pour n'importe quelle chaîne pour des langages tels que $a^n b^n$, $a^n b^m c^{n+m}$ et Dyck-1.

Puisque cet objectif aboutissait à une fonction d'erreur non différentiable, Lan et al. (2022) ont utilisé la neuroévolution pour naviguer l'espace des hypothèses, en faisant évoluer des cellules RNN de forme libre. Puisque nous nous focalisons dans ce travail sur l'objectif, nous laissons ici de côté l'algorithme de recherche et utilisons une seule architecture fixe (LSTM) pour laquelle une cible théorique est connue. Nous inspectons ensuite l'effet de la fonction objectif sur les configurations potentielles du réseau.

Plus largement, des travaux empiriques utilisant les RNN pour l'apprentissage de la grammaire artificielle ont été menés au moins depuis l'introduction des

RNN simples dans Elman (1990). Les travaux théoriques concernant la puissance de calcul théorique des RNN remontent au moins à Siegelmann et Sontag (1992), qui ont montré que les RNN sont Turing-complets sous certaines hypothèses permissives (précision d'activation infinie et temps d'exécution illimité). Le succès empirique des ANN dans le domaine pratique du traitement du langage naturel (NLP) a conduit à un intérêt récent pour la puissance théorique des RNN dans des conditions pratiques, principalement le traitement en temps réel et la précision finie (Weiss et al. 2018, Merrill et al. 2020). D'autres œuvres récentes ont appliqué des méthodes similaires à l'architecture du Transformer (voir aperçu dans Strobl 2023).

En termes de résultats empiriques, des travaux depuis Elman (1990) ont entraîné des réseaux de neurones artificiels à reconnaître les langages formels et ont le plus souvent testé leur capacité de généralisation en utilisant des longueurs et des profondeurs de chaînes de caractères inédites (Gers et Schmidhuber 2001, Joulin et Mikolov 2015, entre autres).

Lan et al. (2023) donnent un aperçu de ces travaux ; ils montrent que le point commun de ces travaux est l'incapacité à généraliser au-delà d'une certaine longueur testée. Lan et al. (2023) fournissent également une référence standardisée pour l'apprentissage des langages formels et constatent que les RNN formés pour optimiser les fonctions d'erreur standard ne parviennent pas à bien généraliser à partir de quantités raisonnablement petites de données, tandis qu'une variante de RNN formée pour minimiser MDL Lan et al. (2022) est capable de

généraliser nettement mieux (potentiellement à l'infini).

L'application du critère MDL aux réseaux de neurones artificiels remonte également au moins au début des années 1990 (voir Schmidhuber (1997) pour un aperçu des premières tentatives dans ce domaine, et Lan et al. (2022) pour un examen des travaux plus récents.) Hinton et VanCamp (1993) ont minimisé la longueur de codage des poids ainsi que l'erreur de prédiction, tout en laissant l'architecture fixe.

Hochreiter et Schmidhuber (1994) ont fourni un algorithme qui recherche les réseaux qui se situent à des « minima plats » – des régions de l'espace des paramètres où l'erreur reste relativement similaire ; cette préférence reçoit une justification MDL.

Zhang et Muhlenbein (1993) ont utilisé un algorithme génétique pour rechercher des architectures de réseau minimisant un objectif MDL, en utilisant un codage de poids similaire à la régularisation L2. Schmidhuber (1997) a présenté un algorithme pour découvrir des réseaux qui optimisent une métrique de complexité basée sur le temps d'exécution, qui est liée au MDL (complexité de Levin).

En fin de compte, nous présentons une comparaison entre les techniques courantes de prévention de surajustement, parmi lesquelles certaines qui assimilent la simplicité à la grandeur scalaire, et l'objectif MDL accompagné d'un schéma de codage pour les paramètres de réseau qui favorise une notion intuitive de simplicité.

Combiné avec un LSTM construit manuellement qui reconnaît le langage

formel $a^n b^n$ de manière optimale, nous sommes arrivés à mesurer les valeurs de l'objectif d'une solution optimale et à vérifier si elles s'alignent sur les points optimaux de la fonction d'erreur. Ce n'est que lorsque nous avons utilisé MDL que le réseau optimal s'est aligné sur le minimum de l'erreur. Pour les autres fonctions d'erreur, les réseaux situés aux points minimaux avaient des performances loin d'être optimales. L'utilisation de méta-heuristiques telles que Early Stopping a atténué dans une certaine mesure le surapprentissage, mais n'a toujours pas conduit à une solution parfaitement générale.

Nous interprétons ces résultats comme un indicateur que l'incapacité des ANN à converger vers des solutions optimales dont il est prouvé qu'elles existent n'est pas accidentelle, mais plutôt une propriété inhérente et pathologique de la manière dont les modèles actuels sont entraînés. Cela se rajoute à une liste croissante d'échecs de généralisation dans l'apprentissage des langages formels, ainsi qu'à des tâches plus complexes en langage naturel.

Nous nous sommes concentrés sur les RNN, principalement parce qu'ils se prêtent facilement à la construction et à l'inspection manuelles. Cependant, nous ne voyons aucune raison a priori pour laquelle nos résultats ne s'étendraient pas à d'autres architectures telles que les Transformers ou les réseaux convolutifs, étant donné la généralité du principe MDL et le fait qu'il s'est avéré bénéfique dans de divers domaines et tâches d'apprentissage, y compris pour les phénomènes linguistiques (voir Stolcke 1994, Grunwald 1996, de Marcken 1996 et Rasin et al. 2021, entre autres).

Chapitre 4

Les linguistes dits "génératifs" argumentent que les humains sont nés avec des biais non triviaux, cette position reposant sur des cas où les connaissances linguistiques des locuteurs vont au-delà de ce qui paraît justifié par les données auxquelles ils ont été exposés. Cela peut également être exprimé de la manière suivante : Si les humains parviennent systématiquement à une connaissance à partir de certaines données, alors que des modèles dits "linguistiquement neutres" exposés aux mêmes données n'y parviennent pas, nous pouvons conclure que les humains ne sont pas linguistiquement neutres : ils arrivent préparés à la tâche de l'acquisition du langage.

Les raisonnements de ce genre sont connus sous le nom de L'argument de la pauvreté du stimulus (The Argument From the Poverty of the Stimulus, APS), et depuis son introduction par Noam Chomsky il y a plus de 50 ans, il était au cœur de l'étude de la capacité linguistique humaine. Dans cet article nous nous concentrerons sur un APS concernant le mouvement wh, alors que d'autres APS divers ont été discutés dans la littérature sur la base d'une série de phénomènes empiriques tels que la "one-substitution" (introduite dans Baker 1978), l'inversion sujet-auxiliaire (introduit dans Chomsky 1971) et Plurals in Compounds (introduits dans Gordon 1985).

Les APS que nous venons de mentionner (ainsi que d'autres) ont été considérées comme un argument en faveur des biais innés non-triviaux chez les humains. Par exemple, l'APS de l'inversion sujet-auxiliaire a été considérée

comme soutenant un biais inné en faveur des transformations hiérarchiques par rapport aux transformations linéaires. L'APS du mouvement *wh* dont nous discutons ci-dessous soutiendra de la même manière un biais complexe dont un modèle linguistiquement neutre n'est pas censé s'en emparer. Il en va de même pour les autres APS dans la littérature. En cela, ces APS vont au-delà de l'observation initiale selon laquelle les enfants peuvent produire et comprendre une infinitude de phrases après être exposé à seulement un nombre fini de phrases (Chomsky 1957, p. 15). Même si généraliser à partir de données finies à un langage infini n'est peut-être pas entièrement trivial, cela est néanmoins à la portée de la plupart des algorithmes d'apprentissage. Et surtout, cette capacité n'implique aucun biais qu'un apprenant linguistiquement neutre n'est pas censé avoir.

Si l'APS a joué un rôle central dans le raisonnement linguistique, il a également généré de nombreuses controverses. Contester une APS donnée nécessite de remettre en question soit les connaissances acquises par les humains, soit les informations dont dispose l'enfant. C'est ces dernières qui sont souvent remises en question : il est extrêmement difficile d'évaluer quelles informations exactes sont disponibles pour l'enfant pendant la période de temps concernée (souvent des années d'apprentissage) et il est très difficile de dire ce qu'un modèle général et linguistiquement neutre en ferait.

On peut essayer de rechercher des éléments de preuve qui semblent pertinents pour les connaissances en jeu — par exemple, comme cela a été fait pour le cas de l'inversion sujet-auxiliaire en anglais par Legate et Yang (2002) — mais

comme le notent Lewis et Elman (2001), Perfors et al. (2011), parmi d'autres, cette méthodologie risque de sous-estimer les informations disponibles : même si nous ne parvenons pas à trouver les preuves que nous cherchons, un modèle général pourrait être en mesure de tirer parti d'autres sources d'informations. Cette méthodologie risque également de surestimer les informations disponibles : même si nous trouvons plusieurs exemples des preuves que nous recherchons, un apprenant généraliste pourrait traiter ces exemples comme du bruit et ne pas parvenir à tirer la conclusion que nous attendons intuitivement. En l'absence d'un véritable modèle qui est capable d'utiliser les informations disponibles dans l'ensemble d'un corpus, il est tout simplement très difficile d'estimer si les données soutiennent les connaissances considérées.

Comment alors raisonner sur l'information dont dispose l'enfant et se demander si elle suffit à favoriser l'acquisition d'un phénomène donné par un modèle linguistiquement neutre ? Dans un monde idéal, on prendrait: (a) un modèle suffisamment puissant, que l'on peut considérer comme non-biaisé en faveur des connaissances pertinentes ; (b) entraîner ce modèle à partir d'un corpus qui correspond au stimulus linguistique que reçoivent les enfants ; et (c) vérifier si le modèle a effectivement acquis les phénomènes considérés.

Dans un tel monde idéal, on pourrait potentiellement travailler avec un algorithme d'induction pour des grammaires non-restreintes, ou avec un langage de programmation général tel que Python. Ces formalismes sont capables de représenter les genres de phénomènes considérés par les linguistes, et à la fois

peuvent être considérés en tant que linguistiquement neutres. Dans notre cas, alors que les grammaires non-restreintes et les programmes Python peuvent facilement représenter l'équivalent du mouvement wh, y compris les subtilités des îles, rien dans l'un ou l'autre cadre ne semble favoriser a priori de telles représentations. On peut bien sûr envisager d'autres cadres de représentation, y compris des formalismes moins puissants (par exemple les grammaires hors contexte) à condition qu'ils puissent toujours représenter les connaissances linguistiques mais ne soient pas biaisés en leur faveur. Il faudrait toujours s'assurer que l'algorithme d'apprentissage lui-même ne biaise pas le modèle pour ou contre les connaissances linguistiques, mais cela peut être fait de différentes manières, par exemple en utilisant un modèle bayésien a priori linguistiquement neutre. Après un entraînement à partir d'un corpus développementalement vraisemblable, correspondant à quelques années d'expérience linguistique humaine, les connaissances acquises par l'algorithme peuvent alors être directement inspectées à l'étape (c).

Dans la réalité actuelle, la combinaison des étapes (a) à (c) est actuellement impossible. Pendant de nombreuses années, la combinaison de (a) et (b) constituait déjà un obstacle majeur. Les algorithmes généraux d'induction de programmes du type que nous venons de mentionner, par exemple, abordent (a) mais échouent sur (b), car ils sont limités à de très petits corpus de formation. À l'autre extrémité de l'échelle, les modèles n-grammes peuvent facilement être entraînés sur de très grands corpus, répondant ainsi à (b), mais leur capacité de représentation est

beaucoup trop limitée pour capturer ou même approximer de manière adéquate des connaissances linguistiques telles que le mouvement wh. D'autres modèles, tels que les grammaires hors-contexte probabilistes, se situent entre ces deux extrémités mais ont toujours du mal à combiner (a) et (b) lorsqu'il s'agit de phénomènes tels que le mouvement wh.

Le défi de l'évaluation des informations dont dispose l'enfant est devenu moins un obstacle ces derniers temps, avec l'avènement des grands modèles de langage (Large Language Models, LLM). Ces modèles, qui s'appuient sur des architectures modernes de réseaux de neurones artificiels, ne répondent pas encore pleinement aux points (a) à (c) — une question qui a été abordée dans la littérature récente et sur laquelle nous reviendrons ci-dessous — mais ils peuvent être entraînés à partir de très grands corpus et réussissent généralement assez bien à acquérir des dépendances séquentielles. Cela a permis à une littérature importante et croissante d'utiliser ces modèles pour poser des questions concernant l'apprentissage des connaissances linguistiques des LLM, souvent avec une référence spécifique à l'APS. Ces travaux commençant par Linzen et al. (2016) et incluant Bernardy et Lappin (2017), Chowdhury et Zamparelli (2018), Gulordava et al. (2018), Kuncoro et al. (2018), Marvin et Linzen (2018), Wilcox et al. (2018, 2019, 2023), Bhattacharya et van Schijndel (2020), Chaves (2020), Warstadt et al. (2020), Huebner et al. (2021), Ozaki et al. (2022), et Yedetore et al. (2023), entre autres, qui ont examiné la préférence des LLM au sein de paires minimales. Nous nous concentrons ici sur l'application des LLM au domaine du mouvement wh, à

la suite de Wilcox et al. (2018, 2019, 2023), Chowdhury et Zamparelli (2018), Bhattacharya et van Schijndel (2020), Chaves (2020), Warstadt et al. (2020) et Ozaki et al. (2022). En particulier, nous examinons la conclusion de Wilcox et al. (2023 ; WFL) que les modèles actuels réfutent un APS dans ce domaine : un APS qui dit que les données ne sont pas suffisamment riches pour permettre à un modèle général d'acquérir le mouvement wh.

L'APS a depuis longtemps été au cœur du raisonnement des linguistes sur l'innéité. Il a toujours été difficile, cependant, d'estimer la quantité d'informations qu'un modèle linguistiquement neutre pourrait espérer extraire de données en pratique. Les réseaux de neurones artificiels modernes promettent de changer cela, et leurs connaissances et leur apprentissage linguistiques ont fait l'objet de recherches dans une littérature croissante. Nous nous concentrons ici sur WFL, qui utilise les LLM pour affirmer que le stimulus est suffisamment riche en matière de mouvement wh et que cela démantèle l'APS dans ce domaine. Nous montrons que cette conclusion est prématurée : en examinant les Parasitic Gaps et le mouvement Across-The-Board, nous montrons que plusieurs ANN ne parviennent pas à atteindre une approximation satisfaisante du phénomène du mouvement wh.

Est-il possible que, à l'avenir, un modèle linguistiquement neutre réussisse là où les modèles que nous avons examinés ont échoué ? Bien sûr. Comme nous en discutons, les modèles actuels sont trop opaques et trop mal compris (et les corpus de formation actuels sont trop irréalistes sur le plan du développement) pour

trancher définitivement la question de savoir si l'APS pour le mouvement wh est valable. Nous notons cependant que les architectures que nous considérons réussissent généralement à approximer de nombreux autres aspects des phénomènes linguistiques et que nous évaluons les modèles en utilisant un critère de réussite extrêmement clément. Et certains modèles ont reçu des apports linguistiques très généreux, dans certains cas plusieurs ordres de grandeur au-delà de ce que les enfants humains reçoivent. Étant donné qu'aucun des LLM n'a atteint une approximation adéquate pour les exemples relativement simples que nous avons considérés – et étant donné qu'au moins un réseau semble capable d'améliorer son approximation lorsqu'il est entraîné sur un corpus biaisé favorablement – nous trouvons plus probable que le stimulus soit tout simplement trop faible pour justifier l'acquisition des aspects pertinents de la connaissance à partir d'un corpus qui est même vaguement réaliste sur le plan du développement par un apprenant linguistiquement neutre. Et si cela s'avère être le cas, la connaissance de ces aspects par les locuteurs adultes est la preuve que les enfants sont de manière innée équipé d'un algorithme d'apprentissage linguistique biaisé pour l'apprentissage des langues telles que nous les connaissons.

RÉSUMÉ

L'objectif de cette thèse est de déterminer comment les réseaux de neurones artificiels (RNA) diffèrent des humains en termes de capacités de généralisation, et comment alors réduire les différences potentielles. Les RNA ont connu une croissance fulgurante ces dernières années, à une croissance à la fois en termes de leur succès et de leur taille. L'hypothèse principale de cette thèse est qu'en imposant des contraintes de simplicité rationnelles, de plus petits RNA pourraient être en mesure d'effectuer de meilleures généralisations.

MOTS CLÉS

Réseaux de neurones artificiels ; Modèles de langage ; Longueur de description minimale ; L'argument de la pauvreté du stimulus.

ABSTRACT

The goal of the thesis is to discover how Artificial Neural Networks (ANNs) differ from humans in terms of generalization capabilities, and how one might bridge this gap. ANNs have exploded in recent years, both in terms of success and of their own sizes. The main proposal in the thesis is that by imposing rational simplicity constraints, smaller ANNs might be able to achieve better generalizations.

KEYWORDS

Artificial neural networks ; Language models ; Minimum Description Length ; The Argument from the Poverty of the Stimulus.