LSTMS Compose — and Learn — Bottom-Up

LSTMS Compose—and Learn—Bottom-Up

Naomi Saphra
University of Edinburgh
n.saphra@ed.ac.uk

Adam Lopez
University of Edinburgh
alopez@inf.ed.ac.uk

Abstract

Recent work in NLP shows that LSTM language models capture hierarchical structure in language data. In contrast to existing work, we consider the *learning* process that leads to their compositional behavior. For a closer look at how an LSTM's sequential representations are composed hierarchically, we present a related measure of Decompositional Interdependence (DI) between word meanings in an LSTM, based on their gate interactions. We connect this measure to syntax with experiments on English language data, where DI is higher on pairs of words with lower syntactic distance. To explore the inductive biases that cause these compositional representations to arise during training, we conduct simple experiments on synthetic data. These synthetic experiments support a specific hypothesis about how hierarchical structures are discovered over the course of training: that LSTM constituent representations are learned bottom-up, relying on effective representations of their shorter children, rather than learning the longer-range relations independently from children.

1 Introduction

For years the LSTM dominated language architectures. It remains a popular architecture in NLP, and unlike Transformer-based models, it can be trained on small corpora (Tran et al., 2018).[1] Abnar et al. (2020) even found that the recurrent inductive biases behind the LSTM's success are so essential that distilling from them can improve the performance of fully attentional models. However, the reasons behind the LSTM's effectiveness in language domains remain poorly understood.

A Transformer can encode syntax using attention (Hewitt and Manning, 2019), and some LSTM variants explicitly encode syntax (Bowman et al., 2016; Dyer et al., 2016). So, the success of these models is partly explained by their ability to model syntactic relationships when predicting a word. By contrast, an LSTM simply scans a sentence from left to right, accumulating meaning into a hidden representation one word at a time, and using that representation to summarize the entire preceding sequence when predicting the next word. Yet we have extensive evidence that trained LSTMs are also sensitive to syntax. For example, they can recall more history in natural language data than in similarly Zipfian-distributed n-gram data, implying that they exploit linguistic structure in long-distance dependencies (Liu et al., 2018). Their internal representations appear to encode constituency (Blevins et al., 2018; Hupkes and Zuidema, 2018) and syntactic agreement (Lakretz et al., 2019; Gulordava et al., 2018). In this paper, we consider how such representations are learned, and what kind of inductive bias supports them.

To understand how LSTMs exploit syntax, we use **contextual decomposition** (CD; Section 2.1), a method that computes how much the hidden representation of an LSTM depends on particular past span of words. We then extend CD to **Decompositional Interdependence** (DI; Section 2.2), a measure of interaction between spans of words to produce the representation at a particular timestep. For example, in the sentence "Socrates asked the student trick questions", we might expect the hidden representation of the LSTM at the word "questions" to interact primarily with its syntactic head "asked", and less with the direct object "the student". If so, then an LSTM could be seen as implementing compositional *localism* (Hupkes et al., 2020): if a hidden representation encodes

[1] As evidence of the ongoing popularity of LSTMs in NLP, a Google Scholar search restricted to `aclweb.org` since 2019 finds 191 citations to the original LSTM paper (Hochreiter and Schmidhuber, 1997) and 242 citations to the original Transformer paper (Vaswani et al., 2017).

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 2797–2809
November 16 - 20, 2020.

meaning, then this meaning is composed from local syntactic relationships. Our experiments on syntactically-parsed corpora (Section 3) illustrate this property — interdependence decreases with syntactic distance, stratified by surface distance.

We then turn to a hypothesis about how such representations are learned. Using a simple synthetic corpus (Section 4.2), we allow LSTMs to learn to represent short sequences before they learn longer sequences that are dependent on them. Our goal is to then illustrate how they *use* representations of short sequences in order to learn longer dependencies—if these smaller constituents are unfamiliar, LSTMs learn more slowly. Further experiments (Section 4.3.1) isolate hierarchical behavior from other factors causing local relations to be learned first, indicating that the model tends to build a subtree from its smaller constituents. We conclude that LSTMs *compose* hierachically because they *learn* bottom-up.

2 Methods

Our DI measure is a natural extension of Contextual Decomposition (CD; Murdoch et al., 2018), a tool for analyzing the representations produced by LSTMs. To conform with Murdoch et al. (2018), our English language experiments use a one layer (400-dim) LSTM, with inputs taken from an embedding layer and outputs processed by a softmax layer.

2.1 Contextual Decomposition

We now will provide a blackbox explanation of CD, the groundwork for our DI. Let us say that we need to determine when our language model has learned that "either" implies an appearance of "or" later in the sequence—a convenient test used since at least Chomsky (1956). We consider an example sentence, "*Either* Socrates is mortal *or* not". Because many nonlinear functions are applied in the intervening span "Socrates is mortal", it is difficult to directly measure the influence of "either" on the later occurrence of "or". To dissect the sequence and understand the impact of individual elements in the sequence, we could employ CD.

CD is a method of looking at the individual influences that words and phrases in a sequence have on the output of a recurrent model. Illustrated in Figure 1, CD decomposes the activation vector produced by an LSTM layer into a sum of relevant and irrelevant parts. The **relevant** part is the exclusive contribution of the set of words **in focus**, i.e., a set of words whose impact we want to measure. We denote this set of words as β. The **irrelevant** part includes the contribution of all words not in that set (denoted $\bar{\beta}$) as well as **interactions** between the relevant and irrelevant words (denoted $\beta \bowtie \bar{\beta}$). For an output hidden state vector h^t, CD will decompose it into two vectors: the relevant h^t_β, and irrelevant $h^t_{\bar{\beta};\beta\bowtie\bar{\beta}}$, such that:

$$h \approx h^t_\beta + h^t_{\bar{\beta};\beta\bowtie\bar{\beta}} \quad (1)$$

This decomposition of the hidden state is based on individual Shapley decompositions of the gating mechanisms themselves, as detailed in Appendix A.

Because the individual contributions of the items in a sequence interact in nonlinear ways, this decomposition is only an approximation and cannot exactly compute the impact of a specific word or words on the label predicted. CD linearizes hidden states with low approximation error, but the presence of slight nonlinearities in the interactions between components forms the basis for our measure of Decompositional Interdependence later on.[2]

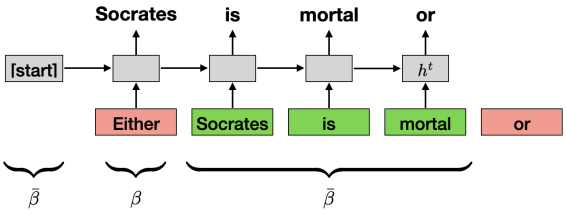


Figure 1: CD uses linear approximations of gate operations to linearize the sequential application of the LSTM module. CD produces the vector h^t_β isolating the contribution of "Either" to the vector h^t predicting "or", as well as producing the irrelevant contribution $h^t_{\bar{\beta};\beta\bowtie\bar{\beta}}$. The irrelevant contribution considers both $\bar{\beta}$ and its interactions with β. In our figures, red will represent matched tokens and green the intervening span of tokens through which information must pass to predict the match.

[2] In our analyses, CD yielded mean approximation error $\frac{\|(v^t_\beta + v^t_{\bar{\beta};\beta\bowtie\bar{\beta}}) - v\|}{\|v\|} < 10^{-5}$ at the logits. However, this measurement misses another source of approximation error: the allocation of credit between β and the interactions $\beta\bowtie\bar{\beta}$. Changing the sequence out of focus $\bar{\beta}$ might influence v^t_β, for example, even though the contribution of the words in focus should be mostly confined to the irrelevant vector component. This approximation error is crucial because the component attributed to $\beta\bowtie\bar{\beta}$ is central to our measure of DI.

We can use softmax to convert the relevant logits (the hidden units after a linear transformation) v_β^t into a probability distribution as $P(Y \mid x_\beta) = \text{softmax}(v_\beta^t)$. This allows us to analyze the effect of input x_β on the probability of a later element while controlling for the influence of the rest of the sequence.

2.2 Decompositional Interdependence

Next, we extend CD to focus on nonlinear interactions. We frame compositionality in terms of whether the meanings of a pair of words or word subsets can be treated independently. For example, a "slice of cake" can be broken into the individual meanings of "slice", "of", and "cake", but an idiomatic expression such as "piece of cake", meaning a simple task, cannot be broken into the individual meanings of "piece", "of", and "cake". The words in the idiom likely have higher **Decompositional Interdependence**, or reliance on their interactions to build meaning. Another influence on DI should be syntactic relation; if you "happily eat a slice of cake", the meaning of "cake" does not depend on "happily", which modifies "eat" and is far on the syntactic tree from "cake", but the meaning of "cake" should be more dependent on "slice", which gives context for its part of speech and suggests that it is concrete.[3] We will use the nonlinear interactions in contextual decomposition to analyze the DI between words alternately considered in focus.

Generally, CD considers all nonlinear interactions between the relevant and irrelevant sets of words to fall under $\beta \reflectbox{$\supset$} \bar{\beta}$, the irrelevant contribution, although other allocations of interactions have been proposed (Jumelet et al., 2019). DI uses these nonlinearities to discover how strongly a pair of spans are associated. A fully flat structure for building meaning could lead to a contextual representation that requires memorization of each word, breaking the simplifying assumption at the heart of CD that each word has an independent meaning to be incorporated into the sentence.

Given two interacting sets of words to potentially designate as the β in focus, A, B such that $A \cap B = \emptyset$, we use a measure of DI to quantify

[3]In our natural language experiments, we focus on dependency relations, but the inductive bias we observe is towards broadly hierarchical patterns in which longer relations depend on local constituents. DI analysis of other sources of this latent hierarchical structure, such as idiom, are left to future work.

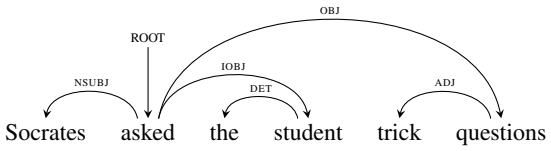


Figure 2: A dependency parsed sentence.

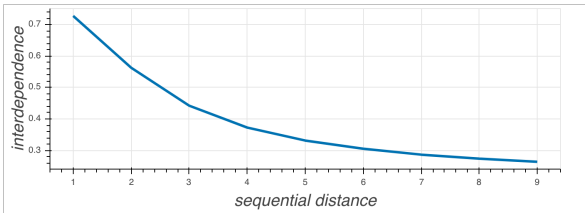


Figure 3: Average DI between word pairs x_l, x_r at different sequential distances $r - l$.

the degree to which $A \cup B$ be broken into their individual meanings. With h_A^t and h_B^t denoting the relevant contributions at the hidden layers of A and B according to CD, and $h_{A\cup B}^t$ as the relevant contribution of $A \cup B$, we compute the magnitude of nonlinear interactions, rescaled to control for the magnitude of the representation:

$$DI^t(A, B) = \frac{\|h_{A\cup B}^t - (h_A^t + h_B^t)\|_2}{\|h_{A\cup B}^t\|_2} \quad (2)$$

This quantity is related to probabilistic independence. We would say that random variables X and Y are independent if their joint probability $P(X, Y) = P(X)P(Y)$. Likewise, the meanings of A and B can be called independent if $h_{A\cup B}^t = h_A^t + h_B^t$. A parallel can also be drawn to Information Quality Ratio (Jetka et al., 2019), a normalized form of mutual information which quantifies information exchanged between two variables against total uncertainty, if we view a decomposed output vector h_β^t as information transmitted from β:

$$IQR^t(A, B) = \frac{H(A, B) - H(A|B) - H(B|A)}{H(A, B)} \quad (3)$$

Note that CD is applied to the representation at a particular timestep, and therefore DI is implicitly an operation that takes three parameters (excluding the sentence): A, B and the timestamp at which to access their representations. However, in order to minimize information degradation over time, we access h^t at the lowest timestep accommodating all spans in focus, $t = \max(\text{idx}(A), \text{idx}(B))$.

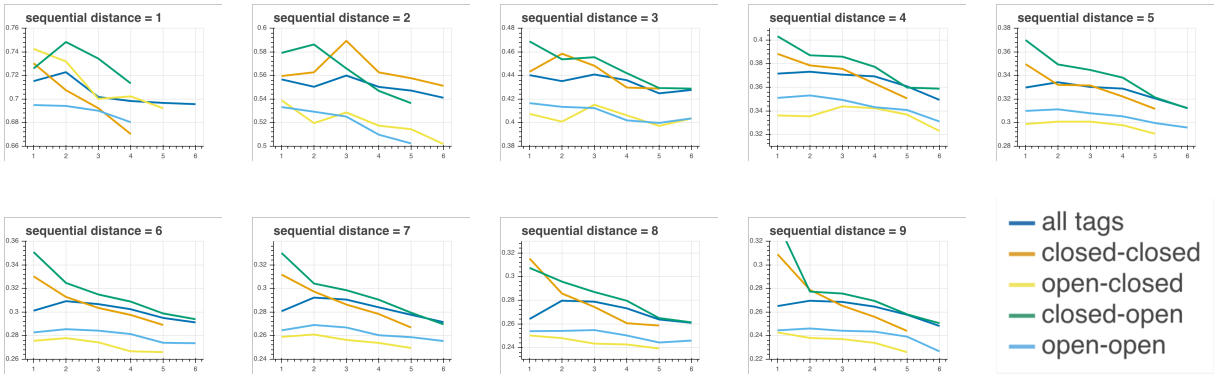


Figure 4: Mean *DI* (y-axis) between word pairs at varying *syntactic distances* (x-axis), stratified by *whether the POS tags are closed or open class* (line color) and by *sequential distance* (plot title). The y-axis ranges differ, but the scale is the same for all plots. Each mean is plotted only if there are at least 100 cases to average.

Concurrently with this work, Chen et al. (2020) also developed a method of studying the interaction between words using Shapley-based techniques like CD. However, their method was based on an assumption of underlying hierarchical structure and therefore unsuitable for the experiments we are about to conduct. Their results nonetheless validate the relationship between feature interaction and syntactic structure.

3 English Language Experiments

We now apply our measure of DI to a natural language setting to see how LSTMs employ bottom-up construction. In natural language, disentangling the meaning of individual words requires contextual information which is hierarchically composed. For example, in the sentence, "Socrates asked the student trick questions", "trick questions" has a clear definition and strong connotations that are less evident in each word individually. However, knowing that "trick" and "student" co-occur is not sufficient to clarify the meaning and connotations of either word or compose a shared meaning.

Here, we consider whether the LSTM observes headedness, by composing meaning between a headword and its immediate modifiers—behavior which a Recurrent Neural Network Grammar (RNNG; Dyer et al., 2016) also learns (Kuncoro et al., 2017). If a standard LSTM learns similar behavior in line with syntax, it is *implicitly* a syntactic language model.

These experiments use language models trained on wikitext-2 (Merity et al., 2016), run on the Universal Dependencies corpus English-EWT (Silveira et al., 2014).

3.1 DI and Syntax

To assess the connection between DI and syntax, we consider the DI of word pairs with different syntactic distances. For example, in Figure 2, "trick" is one edge away from "questions", two from "asked", and four from "the". In Figure 3, we see that in general, the closer two words occur in sequence, the more they influence each other, leading to correspondingly high DI. Therefore we stratify by the sequential distance of words when we investigate syntactic distance.

As synthetic data experiments will show (Section 4), phrase frequency and predictability play a critical role in determining DI (although we found raw word frequency shows no clear correlation with DI in English). In Figure 4, we control for these properties through stratifying by open and closed POS tag class. Open class POS tags frequently accept new words (e.g., nouns and adjectives), whereas closed class tags are mostly consistent historically (e.g., determiners and prepositions). These classes vary in their predictability in context; for example, determiners are almost always soon followed by a noun, but adjectives appear in many constructions like "Socrates is mortal" where they are not. Irrespective of both sequential distance and POS class, we see broadly decreasing trends in DI as the syntactic distance between words increases, consistent with the prediction that syntactic proximity drives DI. This pattern is clearer as words become further apart in the sequence, likely due to the absence of localized non-syntactic influences such as priming effects.

This behavior shows a tendency towards **hierarchical construction** aligned with syntax, wherein the LSTM ties a head's representation together

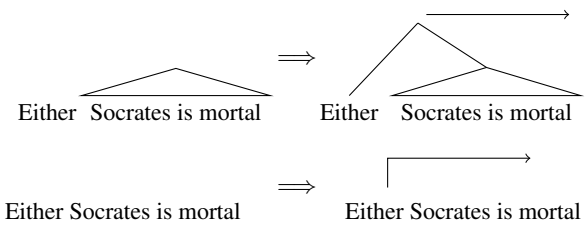


Figure 5: Top: A familiar span (indicated by a triangle illustrating it as a recognizable constituent) is used as a scaffold in its new context, allowing the model to construct a closely interdependent representation for predicting the next word. Bottom: An unfamiliar span cannot be used as a scaffold, so the model is forced to learn the either/or relation independently.

with its child constituents and further associations are less dependent on each other. Similar behavior is the goal of RNNGs and other models which use stack LSTMs (Dyer et al., 2015), which ensure the words in a constituent will be highly interdependent in their shared representation because the constituent will be based on a dictionary lookup for its subtree structure. In an RNNG, this behavior is a result of **bottom-up learning** during training, when the composition operation combines existing tag subtrees into a new lookup key. Our next experiments will illustrate how LSTMs already learn bottom-up implicitly, because they are biased towards the top behavior in Figure 5 when a scaffolding environment is available.

4 Synthetic Experiments

Our next experiments use synthetic data to show how training is bottom-up. LSTM training sees long-range connections discovered after short-range connections; in particular, document-level content topic information is encoded much later in training than local information like part of speech (Saphra and Lopez, 2019).

These experiments explain such learning phases by showing that the training process is inherently compositional due to bottom-up learning.[4] That is, not only are the shorter sequences learned first, but they *form the basis* for longer relations learned over them. For example, the model might learn to represent sequences like "Socrates is mortal" before it can learn to represent the either/or relation around it, building from short constituents to long. This behavior is seen in shift-reduce parsers and their neural derivatives like RNNGs.

Bottom-up training is not a given and must be verified.[5] However, if the hypothesis holds and training builds syntactic patterns hierarchically, it can lead to representations that are built hierarchically at inference time, reflecting linguistic structure, as we have seen. To test the idea of a compositional training process, we use synthetic data that controls for the consistency and frequency of longer-range relations. We find:

1. LSTMs trained with familiar intervening spans have poor performance predicting long distance dependents like "or" without familiar intervening spans (Figure 7). This could be explained by the idea that they never acquire the either/or rule (instead memorizing the entire sequence).

2. But in fact, the either/or rule is acquired *faster* with familiar constituents, as is clear even if the role of "either" is isolated (Figure 8).

3. The poor performance is instead connected to high interdependence between "either" and the intervening span (Figures 9 and 10).

4. Observations (2) and (3) support the idea that acquisition is biased towards bottom-up learning, using the constituent as a scaffold to support the long-distance rule.

4.1 Training Procedure

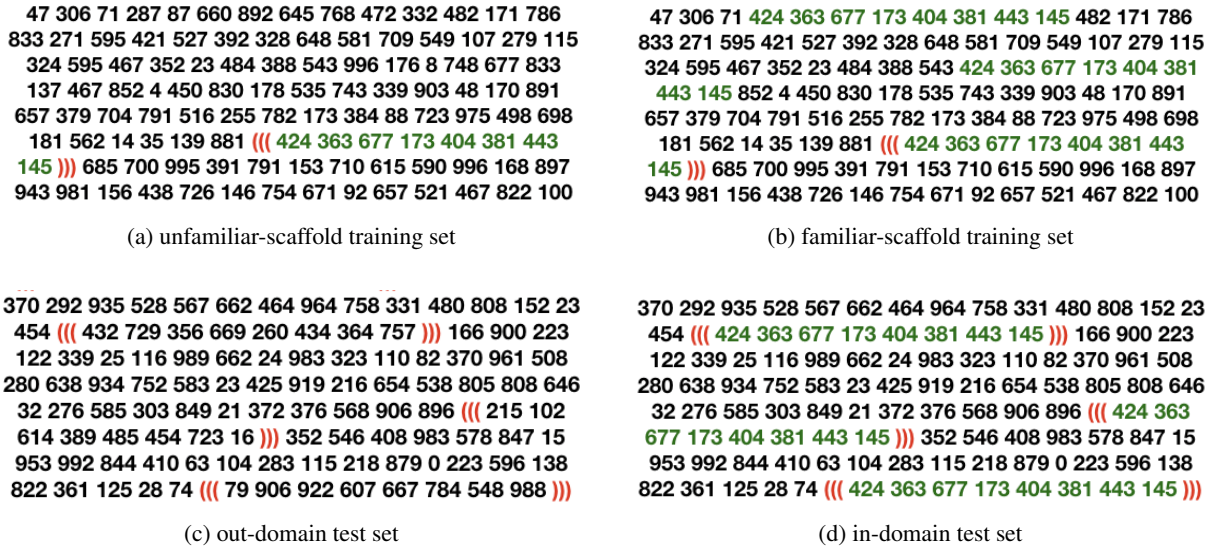
We train our one-layer 200-dim LSTM with a learning rate set at 1 throughout and gradients clipped at 0.25. We found momentum and weight decay to slow rule learning in this setting, so they are not used.

4.2 Long Range Dependencies

First, we describe long-range rules whose acquisition will illuminate compositional learning dy-

[4] Other phenomena contribute but are outside our current focus. First, long-range connections are less consistent (particularly in a right-branching language like English), and will thus take longer to learn (Appendix B. For example, the pattern of a determiner followed by a noun will appear very frequently, as in "the man", while long-range connections like "either/or" are rarer. Second, rarer patterns are learned slowly due to vanishing gradients (Appendix C).

[5] In fact, learning simple rules early on might inhibit the learning of more complex rules through gradient starvation (Combes et al., 2018), in which more frequent features dominate the gradient directed at rarer features. Shorter familiar patterns could slow down the process for learning longer range patterns by trapping the model in a local minimum which makes the long-distance rule harder to reach.



(a) unfamiliar-scaffold training set

(b) familiar-scaffold training set

(c) out-domain test set

(d) in-domain test set

Figure 6: Caricatured train and test datasets for exploring the effect of scaffold familiarity on learning longer distance relations. We have highlighted rule boundaries α and ω in red, and scaffold $q \in Q_k$ in green.

namics. Consider how "either" predicts "or", often interceded by a closed constituent. To learn this rule, a language model must backpropagate information from the occurrence of "or" through the intervening span of words, which we will call a **scaffold**. Perhaps the scaffold is recognizable as a particular type of constituent: in "Either *Socrates is mortal* or not", "or" becomes predictable after a constituent closes. But what if the scaffold is unfamiliar and its structure cannot be effectively represented by the model? For example, if the scaffold includes unknown tokens: "Either *slithy toves gyre* or not". How will the gradient carried from "or" to "either" be shaped according to the scaffold, and how will the representation of that long-range connection change accordingly?

A **familiar** scaffold like "Socrates is mortal" could be used by a bottom-up training process as a short constituent on which to build longer-range representations, so the meaning of "Either" will depend on a similar constituent. Conversely, if training is not biased to be compositional, the connection will be made regardless of the scaffold[6], so the rule will generalize to test data: "either" will always predict "or". This either/or association might *later* develop a dependency on the intervening span due to the nature of the data, but it will initially learn to predict without such scaffolding. We use a synthetic corpus to test these predictions.

In our synthetic corpus, we generate data uniformly at random from a vocabulary Σ. We insert n instances of the long-distance rule $\alpha\Sigma^k\omega$, with scaffold Σ^k of length k, **open symbol** α, and **close symbol** ω, with $\alpha, \omega \notin \Sigma$ (with α as "either" and ω as "or"). Relating to our running example, α stands for "either" and ω stands for "or". We use a corpus of 1m tokens with $|\Sigma| = 1$k types, which leaves a low probability that any scaffold sequence longer than 1 token appears elsewhere by chance.

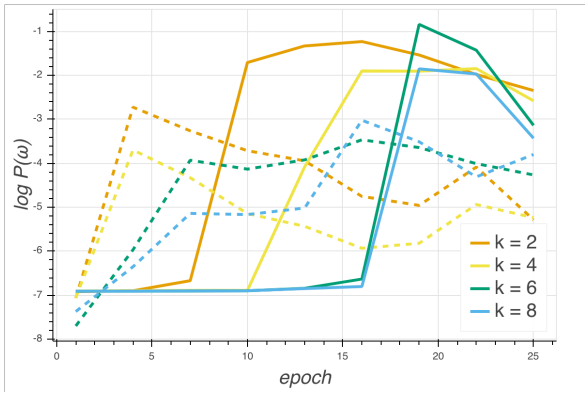
4.3 The Effect of Scaffold Familiarity

To create a dataset of long-range connections with predictable scaffolds, we modify the original synthetic data (Figure 6a) so each scaffold appears frequently outside of the α/ω rule (Figure 6b). The scaffolds are sampled from a randomly generated vocabulary of 100 phrases of length k, so each unique scaffold q appears in the training set 10 times in the context $\alpha q \omega$. This repetition is necessary in order to fit 1000 occurrences of the rule in all settings.

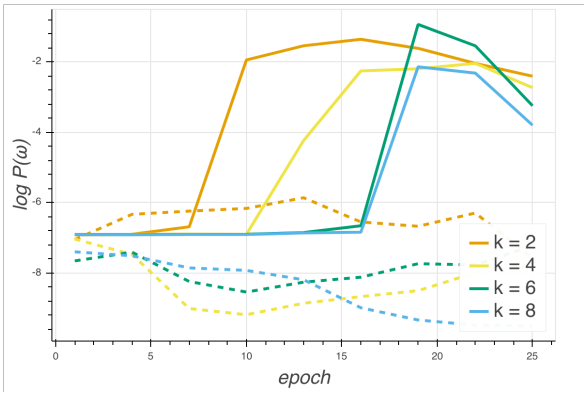
In the **familiar-scaffold setting**, we randomly distribute 1000 occurrences of each scaffold throughout the corpus outside of the rule patterns. Therefore each scaffold is seen often enough to be memorized (see Appendix B). In the original **unfamiliar-scaffold setting**, q appears only as a scaffold, so it is not memorized independently.

We also use two distinct test sets. Our in-domain test set (Figure 6d) uses the same set of scaffolds as the train set. In Figure 7a, the model learns to predict the close symbol faster if the scaf-

[6]Such behavior does reflect another aspect of compositionality, that of *systematicity* (Hupkes et al., 2020).



(a) In-domain scaffold test setting



(b) Random scaffold test setting

Figure 7: Mean marginal target probability of the close symbol in a rule. Solid lines are trained in the unfamiliar-scaffold set, dashed lines on familiar-scaffold. Color is specified by scaffold length (k). Scale of y-axis is matched among graphs.

folds are otherwise memorized. However, this effect may be due to vanishing gradients, discussed below.

These familiar scaffolds do not teach the general long distance dependency rule. If the test set scaffolds are sampled uniformly at random (Figure 6c), Figure 7b shows that the familiar-scaffold training setting never teaches the model to generalize the α/ω rule. For a model trained on the familiar domain, a familiar scaffold is required to predict the close symbol.

Vanishing Gradients: A familiar intervening span is predictably a less effective scaffold, because the familiarity will limit longer distance information due to vanishing gradients. Consider in a simple RNN, as the gradient of the error e^t at timestep t backpropagates k timesteps through the hidden state h:

$$\frac{\partial e^t}{\partial h_{t-k}} = \frac{\partial e^t}{\partial h^t} \prod_{i=1}^{k} \frac{\partial h_{t-i+1}}{\partial h_{t-i}}$$

The backpropagated message is multiplied repeatedly by the gradient at each timestep in the scaffold. If the recurrence derivatives $\frac{\partial h_{i+1}}{\partial h_i}$ are large at some weight, the correspondingly larger backpropagated gradient $\frac{\partial e^t}{\partial h_{t-k}}$ will accelerate descent at that parameter. In other words, an unpredictable scaffold associated with a high error will dominate the gradient's sum over recurrences, delaying the acquisition of the symbol-matching rule. In the case of an LSTM, Kanuparthi et al. (2018) expressed the backpropagated gradient as an iterated addition of the error from each timestep, leading to a similar effect.

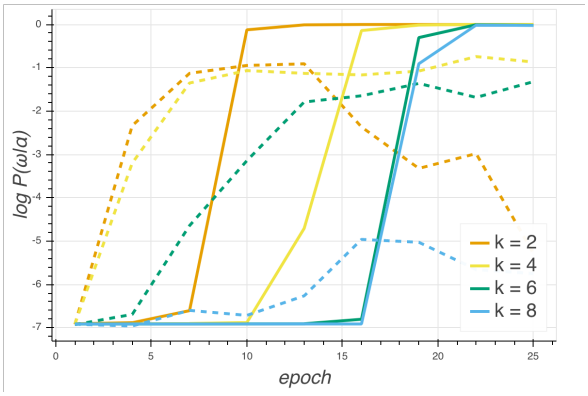


Figure 8: Mean target probability of ω at its correct timestep based on CD with α in focus, on out-domain test set. Solid lines are trained in the unfamiliar-scaffold set, dashed lines on familiar-scaffold.

See Appendix C for confirmation of the difference in gradients between familiar and unfamiliar scaffolds. The speed of acquisition of the dependency rule in a familiar-scaffold training environment therefore has an explanation other than hierarchical composition. Therefore, in order to confirm our proposed compositional bias, we observe the interactions between scaffold and superstructure (long distance dependency) using DI.

4.3.1 Isolating the Effect of the Open-Symbol

Raw predictions in the out-domain test setting appear to suggest that the familiar-scaffold training setting fails to teach the model to associate α and ω. However, the changing domain makes this an unfair assertion: the poor performance may be attributed to wholesale memorization of αq. To illustrate that the rule is learned regardless of training scaffolds, we use CD to isolate the contribu-

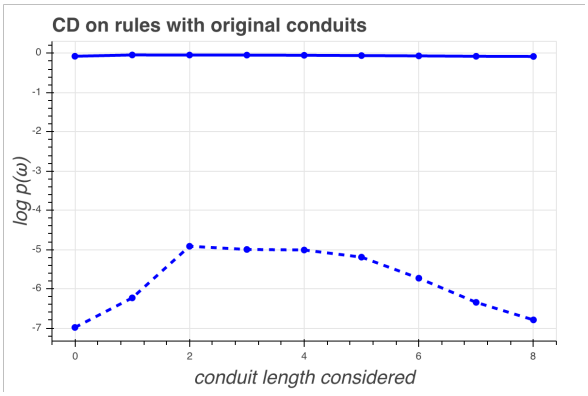


Figure 9: The predicted $P(x_t = \omega | x_{t-k} \dots x_{t-k+i})$ according to CD, varying i as the x-axis and with $x_{t-k} = \alpha$ and $k = 8$. Solid lines are trained in the unfamiliar-scaffold set, dashed lines on familiar-scaffold.

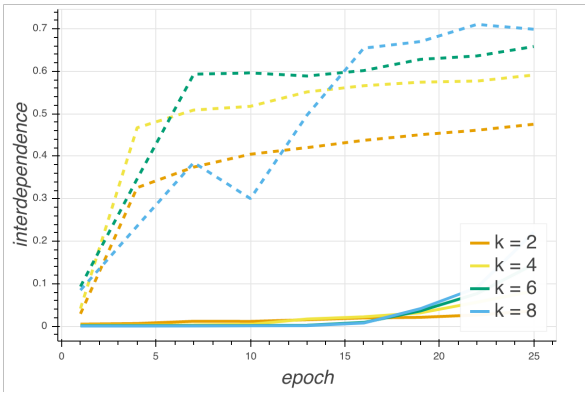


Figure 10: Mean DI(α, scaffold) on the in-domain test set. Solid lines are trained in the unfamiliar-scaffold set, dashed lines on familiar-scaffold.

tions of the open symbol in the out-domain test setting (Figure 8). Furthermore, we confirm that the familiar-scaffold training setting enables earlier acquisition of this rule.

To what, then, can we attribute the failure to generalize out-domain? Figure 9 illustrates how the unfamiliar-scaffold model predicts the close symbol ω with high probability based only on the contributions of the open symbol α. Meanwhile, the familiar-scaffold model probability increases substantially with each symbol consumed until the end of the scaffold, indicating that the model is relying on interactions between the open symbol and the scaffold rather than registering only the effect of the open symbol. Note that this effect cannot be because the scaffold is more predictive of ω. Because each scaffold appears frequently outside of the specific context of the rule in the familiar-scaffold setting, the scaffold is *less* predictive of ω based on distribution alone.

These results indicate that predictable patterns play a vital role in shaping the representations of symbols around them by composing in a way that cannot be easily linearized as a sum of the component parts. In particular, as seen in Figure 10, the DI between open symbol and scaffold is substantially higher for the familiar-setting model and increases throughout training. Long-range connections are not learned independently from scaffold representations, but are *built compositionally* using already-familiar shorter subsequences as scaffolding.

5 Discussion & Related Work

Humans learn by memorizing short rote phrases and later mastering the ability to construct deep syntactic trees from them (Lieven and Tomasello, 2008). LSTM models learn by backpropagation through time, which is unlikely to lead to the same *inductive biases*, the assumptions that define how the model generalizes from its training data. It may not be expected for an LSTM to exhibit similarly compositional learning behavior by building longer constituents out of shorter ones during training, but we present evidence in favor of such learning dynamics.

LSTMs have the theoretical capacity to encode a wide range of context-sensitive languages, but in practice their ability to learn such rules from data is limited (Weiss et al., 2018). Empirically, LSTMs encode the most recent noun as the subject of a verb by default, but they are still capable of learning to encode grammatical inflection from the first word in a sequence rather than the most recent (Ravfogel et al., 2019). Therefore, while inductive biases inherent to the model play a critical role in the ability of an LSTM to learn *effectively*, they are neither necessary nor sufficient in determining what the model *can* learn. Hierarchical linguistic structure may be learned from data alone, or be a natural product of the training process, with neither hypothesis a foregone conclusion. We provide a more precise lens on how LSTM training is itself compositional, beyond the properties of data.

There is a limited literature on compositionality as an inductive bias of neural networks. Saxe et al. (2019) explored how hierarchical ontologies are learned by following their tree structure in 2-layer

feedforward networks. LSTMs also take advantage of some inherent trait of language (Liu et al., 2018) . The compositional training we have explored may be the mechanism behind this biased representational power.

Synthetic data, meanwhile, has formed the basis for analyzing the inductive biases of neural networks and their capacity to learn compositional rules. Common synthetic datasets include the Dyck languages (Suzgun et al., 2019; Skachkova et al., 2018), SPk (Mahalunkar and Kelleher, 2019), synthetic variants of natural language (Ravfogel et al., 2019; Liu et al., 2018), and others (Mul and Zuidema, 2019; Liška et al., 2018; Korrel et al., 2019). Unlike these works, our synthetic task is not designed primarily to test the biases of the neural network or to improve its performance in a restricted setting, but to investigate the internal behavior of an LSTM in response to memorization.

Investigations into learning dynamics like ours may offer insight into selecting training curricula. The application of a curriculum is based on the often unspoken assumption that the representation of a complex pattern can be reached more easily from a simpler pattern. However, we find that effectively representing shorter scaffolds actually makes a language model *less* effective at generalizing a long-range rule, as found by Zhang et al. (2018). This less generalizable representation is still learned faster, which may be why Zhang et al. (2017) found higher performance after one epoch. Our work suggests that measures of length, including syntactic depth, may be inappropriate bases for curriculum learning.

6 Future Work

While we hope to isolate the role of long range dependencies through synthetic data, we must consider the possibility that the natural predictability of language data differs in relevant ways from the synthetic data, in which the scaffolds are predictable only through pure memorization. Because LSTM models take advantage of linguistic structure, we cannot be confident that predictable natural language exhibits the same cell state dynamics that make a memorized scaffold promote or inhibit long-range rule learning. Future work could test our findings on the learning process through carefully selected natural language, rather than synthetic, data.

Our natural language results could lead to DI as a structural probe for testing syntax. Such a probe can be computed directly from an LSTM without learning additional parameters as required in other methods (Hewitt and Manning, 2019). In this way, it is similar to the probes that have been developed using attention distributions (Clark et al., 2019). By computing associations naturally through DI, we can even escape the need to augment models with attention just to permit analysis, as Kuncoro et al. (2017).

Some effects on our natural language experiments may be due to the predictable nature of English syntax, which favors right-branching behavior. Future work could apply similar analysis to other languages with different grammatical word orders.

7 Conclusions

Using our proposed tool of Decompositional Interdependence, we illustrate how information exchanged between words aligns roughly with syntactic structure, indicating LSTMs compose meaning bottom-up. Synthetic experiments then illustrate that a memorized span intervening between a long distance dependency promotes early learning of the dependency rule, but fails to generalize to new domains, implying that these memorized spans are used as scaffolding in a bottom-up learning process.

This combination of behaviors is similar to a syntactic language model, suggesting that the LSTM's demonstrated inductive bias towards hierarchical structures is implicitly aligned with our understanding of language and emerges from its natural learning process.

Acknowledgements

We thank Ida Szubert, Annabelle Michael Carrell, Seraphina Goldfarb-Terrant, Craig Innes, Kate McCurdy, Yevgen Matusevych, Andreas Grivas, Nikolay Bogoychev, Sameer Bansal, Matthew Summers, and Denis Emelin for comments on early drafts of this paper.

References

Samira Abnar, Mostafa Dehghani, and Willem Zuidema. 2020. Transferring Inductive Biases through Knowledge Distillation. *arXiv:2006.00555 [cs, stat]*. ArXiv: 2006.00555.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19, Melbourne, Australia. Association for Computational Linguistics.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany. Association for Computational Linguistics.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593, Online. Association for Computational Linguistics.

N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabanian, Aaron Courville, and Yoshua Bengio. 2018. On the Learning Dynamics of Deep Neural Networks. *arXiv:1809.06848 [cs, stat]*. ArXiv: 1809.06848.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. *CoRR*, abs/1602.07776.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

John Hewitt and Christopher D Manning. 2019. A Structural Probe for Finding Syntax in Word Representations. In *NAACL*.

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

Dieuwke Hupkes, Verna Dankers, Elia Bruni, and Mathijs Mul. 2020. Compositionality Decomposed: How do Neural Networks Generalise? (Extended Abstract). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, volume 5, pages 5065–5069. ISSN: 1045-0823.

Dieuwke Hupkes and Willem Zuidema. 2018. Visualisation and 'Diagnostic Classifiers' Reveal how Recurrent and Recursive Neural Networks Process Hierarchical Structure (Extended Abstract). *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5617–5621.

Tomasz Jetka, Karol Nienałtowski, Tomasz Winarski, Sławomir Błoński, and Michał Komorowski. 2019. Information-theoretic analysis of multivariate single-cell signaling responses. *PLOS Computational Biology*, 15(7):e1007132. Publisher: Public Library of Science.

Jaap Jumelet, Willem Zuidema, and Dieuwke Hupkes. 2019. Analysing Neural Language Models: Contextual Decomposition Reveals Default Reasoning in Number and Gender Assignment. *arXiv preprint arXiv:1909.08975*.

Bhargav Kanuparthi, Devansh Arpit, Giancarlo Kerg, Nan Rosemary Ke, Ioannis Mitliagkas, and Yoshua Bengio. 2018. h-detach: Modifying the LSTM Gradient Towards Better Optimization. In *ICLR*.

Kris Korrel, Dieuwke Hupkes, Verna Dankers, and Elia Bruni. 2019. Transcoding compositionally: Using attention to find more generalizable solutions. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 1–11, Florence, Italy. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, Miguel Ballesteros, Graham Neubig, Lingpeng Kong, and Noah A. Smith. 2017. What Do Recurrent Neural Network Grammars Learn About Syntax? In *EACL*.

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.

Elena Lieven and Michael Tomasello. 2008. *Children's first language acquisition from a usage-based perspective*. Routledge.

Adam Liška, Germán Kruszewski, and Marco Baroni. 2018. Memorize or generalize? searching for a compositional rnn in a haystack. *arXiv preprint arXiv:1802.06467*.

Nelson F. Liu, Omer Levy, Roy Schwartz, Chenhao Tan, and Noah A. Smith. 2018. LSTMs Exploit Linguistic Attributes of Data. *arXiv:1805.11653 [cs]*. ArXiv: 1805.11653.

Abhijit Mahalunkar and John Kelleher. 2019. Multi-element long distance dependencies: Using SPk languages to explore the characteristics of long-distance dependencies. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 34–43, Florence. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mathijs Mul and Willem H. Zuidema. 2019. Siamese recurrent networks learn first-order logic reasoning and exhibit zero-shot compositional generalization. *CoRR*, abs/1906.00180.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs. In *ICLR*.

Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. 2019. Studying the inductive biases of rnns with synthetic variations of natural languages. *CoRR*, abs/1903.06400.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. 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 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2019. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546. Publisher: National Academy of Sciences Section: PNAS Plus.

Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Natalia Skachkova, Thomas Trost, and Dietrich Klakow. 2018. Closing brackets with recurrent neural networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 232–239, Brussels, Belgium. Association for Computational Linguistics.

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.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The Importance of Being Recurrent for Modeling Hierarchical Structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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, Melbourne, Australia. Association for Computational Linguistics.

Dakun Zhang, Jungi Kim, Josep Crego, and Jean Senellart. 2017. Boosting Neural Machine Translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 271–276, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J. Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. 2018. An Empirical Exploration of Curriculum Learning for Neural Machine Translation. *arXiv:1811.00739 [cs]*. ArXiv: 1811.00739.

A Details of Contextual Decomposition

For an output hidden state vector h^t, CD will decompose it into two vectors: the relevant h^t_β, and irrelevant $h^t_{\bar{\beta};\beta\bar{\beta}}$, such that:

$$h \approx h^t_\beta + h^t_{\bar{\beta};\beta\bar{\beta}}$$

This decomposed form is achieved by linearizing the contribution of the words in focus at each gate. This is necessarily approximate, because the internal gating mechanisms in an LSTM each employ a nonlinear activation function, either σ or tanh. Murdoch et al. (2018) use a linearized approximation L_σ for σ and linearized approximation $L_{\tanh}$ for $\tanh$ such that for arbitrary input $\sum_{j=1}^N y_j$:

$$\sigma\left(\sum_{j=1}^N y_j\right) = \sum_{j=1}^N L_\sigma(y_j) \tag{4}$$

These approximations are then used to split each gate into components contributed by the previous hidden state h^{t-1} and by the current input x^t, for example the input gate i^t:

$$\begin{aligned} i^t &= \sigma(W_i x^t + V_t h^{t-1} + b_i) \\ &\approx L_\sigma(W_i x^t) + L_\sigma(V_t h^{t-1}) + L_\sigma(b_i) \end{aligned}$$

The linear form L_σ is achieved by computing the Shapley value (Shapley, 1953) of its parameter, defined as the average difference resulting from excluding the parameter, over all possible permutations of the input summants. To apply Formula 4 to $\sigma(y_1 + y_2)$ for a linear approximation of the isolated effect of the summant y_1:

$$L_\sigma(y_1) = \frac{1}{2}[(\sigma(y_1) - \sigma(0)) + (\sigma(y_2 + y_1) - \sigma(y_1))]$$

With this function, we can take a hidden state from the previous timestep, decomposed as $h^{t-1} \approx h^{t-1}_\beta + h^{t-1}_{\bar{\beta};\beta\bar{\beta}}$ and add x^t to the appropriate component. For example, if x^t is in focus, we count it in the relevant function inputs when computing the input gate:

$$\begin{aligned} i^t &= \sigma(W_i x^t + V_t h^{t-1} + b_i) \\ &\approx \sigma(W_i x^t + V_t(h^{t-1}_\beta + h^{t-1}_{\bar{\beta};\beta\bar{\beta}}) + b_i) \\ &\approx [L_\sigma(W_i x^t + V_t h^{t-1}_\beta) + L_\sigma(b_i)] \\ &\quad + L_\sigma(V_t h^{t-1}_{\bar{\beta};\beta\bar{\beta}}) \\ &= i^t_\beta + i^t_{\bar{\beta};\beta\bar{\beta}} \end{aligned}$$

This provides an expression of the approximate input gate as the sum of relevant and irrelevant components. By ignoring the irrelevant components while computing the module output h^t, we produce h^t_β. Thus we linearize and isolate the effect of β.

B The Effect of Rule Frequency and Length

Here, we investigate how the frequency of a rule affects the ability of the model to learn the rule by varying the number of rule occurrences n and the rule length k.

The results in Figure 11 illustrate how a longer scaffold length requires more examples before the model can learn the corresponding rule. We consider the probability assigned to the close symbol according to the contributions of the open symbol, excluding interaction from any other token in the sequence. For contrast, we also show the extremely low probability assigned to the close symbol according to the contributions of the scaffold taken as an entire phrase. In particular, note the pattern when the rule is extremely rare: The probability of the close symbol β as determined by the open symbol α is low but steady, while the probability as determined by the scaffold declines with scaffold length due to the accumulated low probabilities from each element in the sequence.

C Smaller scaffold gradient, faster rule learning

Figure 12 confirms that a predictable scaffold is associated with a smaller error gradient. Because of the mechanics of backpropagation through time next described, this setting will teach the α/ω rule faster.

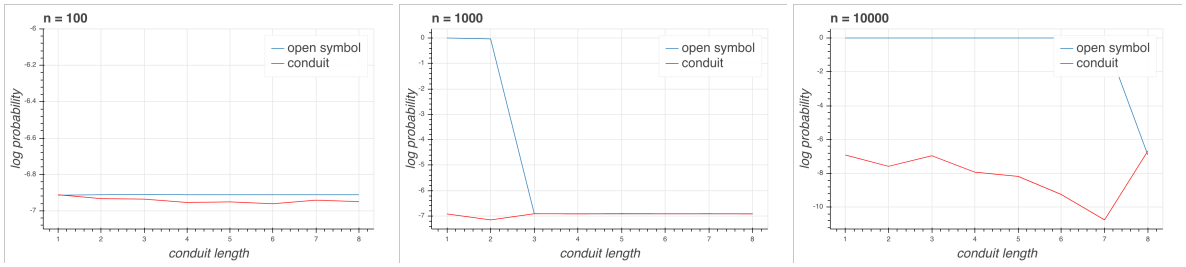


Figure 11: The predicted probability $P(x_t = \omega)$, according to the contributions of open symbol $x_{t-k} = \alpha$ and of the scaffold sequence $x_{t-k+1} \dots x_{t-1}$, for various rule occurrence counts n. Shown at 40 epochs.

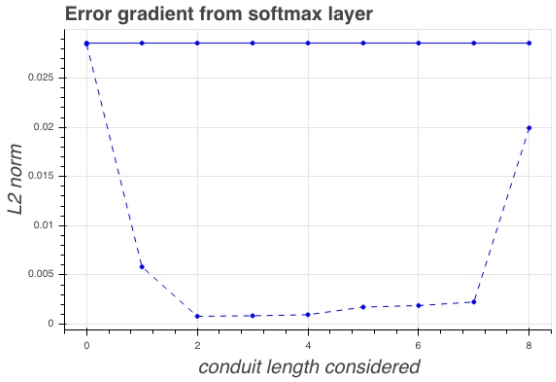


Figure 12: Average gradient magnitude ΔE_{t+-k+d}, varying d up to the length of the scaffold. Solid lines are the unpredictable scaffold setting, dashed lines are the predictable scaffold setting.