Training Dynamics of Neural Language Models

Naomi Saphra

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2025

Abstract

Why do artificial neural networks model language so well? We claim that in order to answer this question and understand the biases that lead to such high performing language models—and all models that handle language—**we must analyze the training process**. For decades, linguists have used the tools of developmental linguistics to study human bias towards linguistic structure. Similarly, we wish to consider a neural network's training dynamics, i.e., the analysis of training in practice and the study of why our optimization methods work when applied. This framing shows us how structural patterns and linguistic properties are gradually built up, revealing more about why LSTM models learn so effectively on language data.

To explore these questions, we might be tempted to appropriate methods from developmental linguistics, but we do not wish to make cognitive claims, so we avoid analogizing between human and artificial language learners. We instead use mathematical tools designed for investigating language model training dynamics. These tools can take advantage of crucial differences between child development and model training: we have access to activations, weights, and gradients in a learning model, and can manipulate learning behavior directly or by perturbing inputs. While most research in training dynamics has focused on vision tasks, language offers direct annotation of its well-documented and intuitive latent hierarchical structures (e.g., syntax and semantics) and is therefore an ideal domain for exploring the effect of training dynamics on the representation of such structure.

Focusing on LSTM models, we investigate the natural sparsity of gradients and activations, finding that word representations are focused on just a few neurons late in training. Similarity analysis reveals how word embeddings learned for different tasks are highly similar at the beginning of training, but gradually become task-specific. Using synthetic data and measuring feature interactions, we also discover that hierarchical representations in LSTMs may be a result of their learning strategy: they tend to build new trees out of familiar phrases, by mingling together the meaning of constituents so they depend on each other. These discoveries constitute just a few possible explanations for how LSTMs learn generalized language representations, with further theories on more architectures to be uncovered by the growing field of NLP training dynamics.

Lay Summary

A lot of recent work has asked how neural networks produce such effective representations of language. Some of this work has asked, "How is linguistic structure encoded in this representation?" Some work has asked, "How does the learning process lead to such effective representations?" We want to ask a combination of these questions: **How does the learning process lead to the encoding of linguistic structure?**

To answer this question, we look at the representations of words over the course of training and we describe how they change. We see that models learn short sequences first, building long sequences out of them, and that therefore the representations contain information about simple word properties long before they contain information about the larger document they came from.

Acknowledgements

Some who know me might find it surprising to learn that I was considered quite bad at math at the age of eight. It seems I hadn't been sufficiently enthusiastic in memorizing my multiplication tables, and the matter wasn't helped by the fact that, in place of any childish charm, I was bestowed with several rather ugly (though blessedly temporary) facial features. Whatever the cause, my math teacher was determined to demote me to a remedial course, having no other recourse to avoid the daily sight of my face and my consistent lack of passion for numerical tables. However, at the required meeting with administration and family, she found herself unable to point to any particular problem I had answered *wrong*, per se. Fortunately, all parties agreed to a compromise in which I instead moved to a more advanced math track, thus preventing my cynicism from polluting her table-memorization sessions any further.

Given that I retained my uncanny ability to repel math teachers well into my teenage years (if not further), I wish to first recognize the list of mentors who encouraged my interest in math. In chronological order: Lauris Khan, who taught me geometry the summer I was 14; Prof. Matthew Blaschko, who taught me how much I needed to know about linear algebra five years later; and Prof. Raman Arora, who taught me I still didn't know anything about linear algebra when I started grad school. Without their collective influence, I'm sure I would have found some tedious but lucrative software engineering job instead.

Another set of people I have to thank are those who have enabled me to use a computer at all for the last several years. Dr. Norman Latov of Weil Medical College at Cornell University and Prof. Hugh Willison of the University of Glasgow are my neurologists on each side of the Atlantic. They have helped in my battle for the dignity of a diagnosis, which is more than many people can hope for.[1] Ryan Hileman[2] and, previously, Ben Meyer[3] have developed the software necessary for me to dictate code and formulae when I couldn't type or write, so thank you both. And thank you, Kami Vaniea and Helen Pain, for contriving to place me in a private office full of plastic dinosaur toys so I could be alone to dictate my work.

At this point, I thank my supervisor, Dr. Adam Lopez (an event which allows the

[1] My gratitude is boundless here, and my potential enumeration of grudges against other, unlisted, neurologists is far longer than any possible screed against math teachers.

[2] https://talonvoice.com/

[3] https://www.voicecode.io/

majority of readers to close this thesis, or at least to skip to the end of my Acknowledgements to see if I wrote anything juicy there). He supported me for years when it was uncertain that I would ever code again, let alone finish this thesis. He always reminded me to take the breaks I needed, even when it was clear that I was actually quite lazy and taking plenty of breaks. He shot down my terrible ideas so they could be reborn phoenix-like from the ashes as less terrible ideas. It is easy to imagine a world in which I had a lesser advisor, who gave up when I was ready to give up. This dissertation would surely not exist in such a world.

Edinburgh University, and especially Adam's group, AGORA,[4] is full of more friends than I can name. But more specifically, it is full of people who took time out to read through parts of this thesis and who should be blamed if it still has typos. I therefore would like to thank by name Ramon, Ida, Craig, Seraphina, Kate, Nik, Yevgen, and Andreas. Outside of my department, a number of friends have supported my work. Matthew Summers was unfailingly kind, an example I have imitated poorly, and helped generate several plot figures in my papers for no good reason other than just being a good friend. Faye Wang made my writing better so I could become extremely famous. Prof. Rachel Rudinger gives me a scavenger hunt for conferences.[5] Annabelle, Jake, and Craig all helped me when I lost the ability to communicate. Mohammad: thanks for being my exopod friend.

Then there are my mentors and collaborators. From Edinburgh: Mohammad Tahaei and Prof. Kami Vaniea. From Google: Eric Altendorf, Dr. Marius Pasca, Dr. Dipanjan Das, Dr. Jasmijn Bastings, Dr. Thibault Sellam, and Dr. Ian Tenney. From Cambridge: Jennifer White and Tiago Pimentel. From FAIR: Dr. Adina Williams. From Johns Hopkins: Prof. Ryan Cotterell, Dr. Adithya Renduchintala, Prof. Sanjeev Khudanpur, and Prof. Raman Arora. From Carnegie Mellon: Prof. Nathan Schneider, Prof. Brendan O'Connor, Prof. David Bamman, Dr. Manaal Faruqui, Prof. Noah Smith, and Dr. Chris Dyer. Thank you to Dr. Catherine Havasi, who never knew me but gave a talk that convinced me I wanted to study NLP when I was a teenager. Thank you to Neal Stephenson and Prof. Douglas Hofstadter, who never knew me but wrote books that convinced me I wanted to study AI and language when I was a teenager.

I also thank my thesis committee: Yonatan Belinkov, who challenged me on causality; and Shay Cohen, who challenged me on linearity. Readers in the distant future, when

[4] Adam's Got an Overdone Recerché Acronym

[5] "Have you seen Rachel Rudinger?"

my thesis continues to be referenced widely, must understand that it was written in a plague year, when all defenses had to be held remote. You, in your future underwater cities, might be used to this, but for us it was a struggle. It was therefore remarkable that they made my viva productive and even enjoyable, and their feedback has certainly improved the completed work.

Of course, there are my parents. I cannot acknowledge them any more eloquently than my mother acknowledges herself, so in her own words: "Who was there to get you back and forth for extra classes? Who was there to fight with that third grade math teacher for you? Who was there to drive you all the way from New York to Pittsburgh to move for undergrad? Yeah, that's right." Thank you so much, David and Phyllis. You put a lot into raising me, but at least now you get to address your daughter as Dr. Saphra; it is, in fact, required of you.

Thank you to my flatmate, Ida Szubert. You proofread more of this thesis than anyone, other than Adam, and not only that—you create the home in which we both work. As I write this, two mice are scurrying across the floor, trapped here by my ill-considered plan to seal their mouse-hole without checking where they were. I am now trying to scare them away with loud music.[6] These are the sort of problem-solving skills I apply in my home life, whereas you fix sweaters, research purchases, and keep tidy. Without you, I surely would have drowned in the bathtub or broken my neck on our unlit staircase long ago.[7][8]

Annabelle Carrell! You were by my side the whole time I wrote. I so want to be by your side for every future accomplishment, yours or mine. I anticipate many incredible accomplishments on your end, because you are clever and driven and smart enough to get out of the farm[9] to wherever you want. We are at a crossroads as I write this—we don't know where you or I will live next, though we can hope and plan. But whatever happens, know this: the plague year was the best year of my life, because I spent it with you.

[6] Specifically: Cat Stevens, Cat Power, and Doja Cat.
[7] Update: Ida helped me chase the mouse out the door. It was exhilarating.
[8] The staircase lights have been repaired.
[9] Not a farm.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Naomi Saphra*)

For Miriam, Raphael, Annabelle. Please clear my inland.

Contents

Chapter 1

Introduction

> "Maybe if I buy some I can learn how to use them," said Milo eagerly as he began to pick through the words in the stall. Finally he chose three which looked particularly good to him—"quagmire," "flabbergast," and "upholstery." He had no idea what they meant, but they looked very grand and elegant.
>
> Norton Juster, *The Phantom Tollbooth*

Language models (LMs), a crucial element of applications that require text generation, have seen many implementations in their decades-long history. The current the era of artificial neural networks has seen a surge in exuberant claims of underlying linguistic competence. However, we still see a lack of accompanying serious investigation into how modern models like LSTMs (Hochreiter and Schmidhuber, 1997b) actually learn.

For years, the LM world was dominated by n-gram models (Shannon, 1948; Rosenfeld, 2000). These models would directly store sequences of words and record their probabilities, holding to a Markov assumption that history was only relevant a predefined distance back. In an n-gram model, we would record a table directly mapping "sentence", "example sentence", and "silly example sentence" to the number of times they were observed in a training corpus. Confronted with n-gram models, it would have been unusual to claim that such a table fundamentally represented syntax.[1] But mod-

[1] This is not to say such claims were not made. A Markov model based on shifting windows (easily implemented with an n-gram lookup table) seems to be the basis of the Sausage Machine, which Frazier and Fodor (1978) provided as a model of human language processing. Later, Wanner (1980) would declare "the Sausage Machine's putative explanation of . . . behavior . . . is simply incorrect", criticizing the over-interpretation of a mechanical model, in a tradition that continues to this day in NLP.

ern language models, from **Transformers** (Vaswani et al., 2017) and other **fully attentional models** (Brown et al., 2020; Liu et al., 2019b; Devlin et al., 2019; Yang et al., 2019) trained on petabytes of internet data, to more lightweight **Long Short Term Memory networks** (LSTMs; Hochreiter and Schmidhuber, 1997b), are frequently subject to such claims (Linzen et al., 2016; Lakretz et al., 2019; Kuncoro et al., 2018; Lu et al., 2020; Vig and Belinkov, 2019; Htut et al., 2019; Vashishth et al., 2019; Wiegreffe and Pinter, 2019; Clark et al., 2019; Voita et al., 2019b; Marecek and Rosa, 2019). Gulordava et al. (2018), for example, suggested that "the ability to capture structural generalizations is an important aspect of what makes the best RNN architectures so good at language modeling"—or even further, "Recurrent neural networks empirically generate natural language with high syntactic fidelity." (Hewitt et al., 2020). For example, LSTMs prefer "*complains*" over "the philosopher the Greeks like *complain*", modeling relative clauses syntactically rather than just using the most recent noun's ("Greeks") inflection (Gulordava et al., 2018).

The design of these opaque modern models does not explicitly encode human linguistic intuitions such as underlying syntactic structure—and yet, their success at language modeling and generation seems to indicate that the trained models encode these intuitions, enough that researchers ask questions about *how*. A variety of methods have therefore been employed to test these models for syntactic structure. Many methods analyze the intermediate representations and internal behavior of language models. Some probe the representations with small classifiers, looking for properties like part of speech implicitly encoded in the vectors that represent particular words (Belinkov et al., 2017; Voita and Titov, 2020). Others consider whether the geometry of these representational spaces reveals the connections between words in a sentence (Hewitt and Manning, 2019). Still other tests consider what words a model pays attention to at each step in the sequence (Voita et al., 2019b; Clark et al., 2019). These methods share a property: they examine the fully trained model, considering only how it encodes language after it has been fully trained. This dissertation moves beyond such a limited analysis, developing an understanding of the model by investigating the entirety of the training process.

1.1 The Lens of Inductive Bias

Often, questions reach beyond a particular set of parameters and behavior, asking questions about the **inductive bias** associated with a model. This term is often nebulous in use, but broadly it refers to the assumptions made by a learning algorithm. This set of assumptions form a resulting model's biases—in particular, those that don't come from the training data itself. No set of biases are universal; for example, biases that support language modeling on English might damage the same model's performance on Chinese. Such trade-offs are a proven limit of learning algorithms, demonstrating a rule known as the No Free Lunch Theorem (Wolpert and Macready, 1997).

Geirhos et al. (2020) describe inductive bias as consisting of four components:

1. **Structure:** The functions chosen to combine parameters and inputs. In this thesis, architectures are centered around an LSTM module, though many modern language models tend to be fully attentional and use Transformers[2].
2. **Experience:** The training data used to train the model.
3. **Goal:** The objective optimized in training. Most neural classifiers use a cross-entropy loss function, but Geirhos et al. (2020) point out that regularizers are often used to prevent overfitting by memorizing training data or using other undesirable heuristics.
4. **Learning:** The optimization process itself. This includes decisions such as whether to use gradient clipping or some implementation of momentum.

In practice, however, we cannot disentangle these factors entirely[3]. An optimizer runs on a loss surface defined by the relationship between goal and structure, as guided by experience. These different factors are selected specifically *expecting* the others, e.g., while we use Adam (Kingma and Ba, 2015) to help optimize an LSTM for language data, it might not help the same architecture to memorize random noise. As Wang (2020) put it, the simultaneous consideration that has gone into choosing a goal, engineering an architecture, and designing an optimizer to match means that every modern neural network is "a consequence of data-driven optimization, engendering the inductive bias—the free lunch is paid for by all the unfit that failed to survive natural selection". Because the aspects of inductive bias have all been optimized jointly,

[2]Throughout the thesis, we note how our methods and results can generalize to attentional models.
[3]We can still try, with dataset manipulation (Section 2.2), ablation tests, or causal analysis.

we have presented empirical results that treat the learning process as a whole, rather than force our expectations onto a particular architecture or corpus.

1.1.1 The Role of Inductive Bias in NLP

Inductive bias is crucial to the performance of language models, or any model handling natural language—indeed, any predictive model at all. Although massive Transformers are the high-performance darlings of the current NLP landscape, recurrent operations like those in the LSTM are still valuable. The smaller the training data, the more crucial inductive bias becomes to prevent the model from memorizing or adopting overly complex heuristics.

Tran et al. (2018) found that tasks which required a model to represent latent hierarchical structure, such as subject-verb agreement (Linzen et al., 2016) and the artificial logic language of Bowman et al. (2015), saw higher performance from RNNs than fully attentional models. They therefore demonstrated the role of the architecture itself in imposing inductive bias.

The architecture is not the only element in inductive bias, however. Abnar et al. (2020) connected inductive bias back to the goal during learning by demonstrating how fully attentional models benefited by learning from the representations produced by recurrent models. Levy and Goldberg (2014) illustrated that the word2vec (Mikolov et al., 2013b) objective and regularizer were identical to a combined objective that could be optimized by matrix factorization, but they could not achieve the same performance with a representation using this equivalent matrix decomposition. This performance gap is likely because the optimizer is essential for implicitly regularizing the word2vec objective, in addition to the goal and architecture.

If any ML domain were oriented towards an investigation into the biases of training processes, it is language. For decades, linguists have asked whether humans have some inherent inductive bias that points us towards linguistic structure (Chomsky and Keyser, 1988), or whether we use a generic neural architecture that gradually learns language the same as it would any other set of rules: through overwhelming input and feedback. This question is not answered by studying how a person processes a sentence after a lifetime of language exposure. Instead, linguists apply crucial tools from the field of developmental linguistics, which studies how human language is acquired over the course of a child's development. Our goal in this thesis is likewise to present

empirical work on the acquisition of language in artificial neural networks.

We claim that in order to understand the biases that lead to high performing language models—and all models that handle language—we must analyze the training process. Studies of how a model optimization regime works in practice, i.e., of *training dynamics*, will reveal how structural patterns and linguistic properties such as syntax are gradually built up hierarchically, revealing *why* neural language models learn so effectively and how they "see" the data.

If we wish to explore questions about how LSTMs learn language, we might be tempted to appropriate methods directly from developmental linguistics. However, in this work we avoid framing artificial neural networks as cognitive models. We instead use specific and mathematical tools for investigating language model training dynamics, such as synthetic datasets and similarity-based model comparisons. These tools can take advantage of crucial differences between child development and model training: we have access to activations, weights, and gradients in a learning model, and can manipulate learning behavior directly or by perturbing inputs. The tools we use often come from computer vision research, as research on training dynamics tends to focus solely on vision tasks, but language has well-documented and intuitive latent hierarchical structures (e.g., syntax and semantics) which make it an ideal domain for exploring the effect of training dynamics on the representation of such structure.

1.2 Structure of Thesis

We claim that training dynamics provide an essential view for understanding language structure in LMs. We support this claim with examples from several case studies, introducing new tools for inspecting neural networks during training. Along with background and commentary, we illustrate these tools, and investigate what they tell us about the development and bias of language models. The novel work in this thesis appears in order from the most conventionally language-motivated ("NLP") to the most conventionally domain-abstracted ("machine learning"). This leads to a chapter structure as follows:

Chapter 2: Background: Language Structure in Models This thesis lies at the intersection of two existing fields. The first is NLP interpretability, which focuses on understanding why text models work by investigating how their behavior expresses

fundamental linguistic properties, in particular syntax and semantics. The interpretation methods we focus on show how these structures result (or fail to result) in the biases of model architectures and in the behavior of trained models.

Chapter 3: Background: Training Dynamics The second field we survey is training dynamics, which aims to describe how a model changes as it trains, often investigating the biases that allow a model to learn effectively. We discuss how a number of works imply that, at varying scales, training happens in phases. We then discuss how the movement of weights during training responds to the underlying structure of input data. Finally, we consider a variety of methods commonly used to investigate learning dynamics and the conclusions about training that they have reached. Some of these methods (particularly SVCCA; Section 2.4.2.1.1) are used directly in this thesis, while others have led to conclusions about the training process that inspire further work here.

Chapter 4: Beyond Diagnostic Classifiers: Probing Language Model Development How does an LSTM LM's acquisition of language information vary, when we compare local syntactic properties with source document information? We discover that conventional diagnostic classifiers, a popular method for assessing the linguistic properties of language models, are not sensitive enough to capture changes during training, so we develop an alternate method. We apply SVCCA, a simple and flexible similarity measurement, to compare the development of LSTM language models predicting the next *word* with LSTM taggers predicting *general categories* (part of speech, semantic tag, and source; which vary in the type of data used for accurate tagging). We find that part-of-speech is learned earlier than topic, indicating that local structure is learned well before long-distance information. This discovery inspired work on hierarchical construction of meaning in Chapter 5. We also find that an LM's recurrent layers become more task-independent over the course of training, while an embedding layer becomes more task-specific later in training. We point out that the tendency to lose shared input structure later in training resembles the predictions of a controversial theory about phase-based learning, the Information Bottleneck Hypothesis.

Chapter 5: Beyond Probing: The Development of Word Interdependence Why do hierarchical structures like syntax tend to emerge in LSTMs? We move from a view of the output representation space as a whole to focus on local interactions between word pairs, asking how an LSTM moves from shorter relations like those re-

quired for POS prediction to longer relations like those required for topic tagging in Chapter 4. We measure interactions between word pairs by extending a recent method called Contextual Decomposition to a measure we call Decompositional Interdependence. Applying this measure in a set of experiments on synthetic languages supports a specific hypothesis about how hierarchical structures are discovered over the course of training: that LSTMs rely on smaller constituents as scaffolding for larger trees, rather than learning longer-range relations (e.g., "either"/"or") independently of their intervening constituents.

Chapter 6: Beyond Words: The Development of Feature Sparsity How does the distribution of vector unit magnitude and neuron importance change over the course of training? Here, we investigate a tendency in LSTMs to change in the sparsity of their gradients and activations over time. We find that frequent input words are associated with sparse activations, while frequent target words are associated with dispersed *activations* but sparse *gradients* (which relate to neuron salience). We find that gradients associated with function words are more sparse than the gradients of content words, even controlling for word frequency—could this sparsity signify a stronger role in defining linguistic structure? These properties change dramatically over time, with some layers beginning dense and growing sparser while others remain stable. We consider whether the gradient sparsification which we observe may be an expression of a compression phase from the Information Bottleneck Hypothesis, similar to the link from Chapter 4.

Chapter 7: Conclusion We discuss recent developments building on the work in this thesis. Based on these developments in addition to our own work, we explore the implications of this thesis and possible future directions.

1.3 Contributions

A Note on Contributions All papers reprinted in this thesis are joint work with my supervisor, Adam Lopez. In all cases, I conceived and implemented the original ideas, and Dr. Lopez helped strengthen these ideas through discussion, especially by challenging them and demanding more specific claims and stronger evidence. The papers themselves were written primarily by me, with Dr. Lopez editing, "killing my darlings" (Quiller-Crouch, 1916), and adding some of the visualizations.

- Understanding Learning Dynamics of Language Models With SVCCA
  - To our knowledge, this is the first in-depth study of learning dynamics of neural language models.
  - We introduce a flexible new probing method based on model similarity, which enables us to compare learned representations across time and across models. We compare the representations produced by language models (word predictors) and tag predictors. This probing method does not require us to have annotated evaluation data, and is efficient to train because it is based on simple matrix factorizations.
  - We find that coarse part of speech is learned first, and topic information is learned last, with fine-grained and semantic information learned in between. Early in training, models targeting different tasks with the same inputs tend to produce similar representations, and then specialize to their tasks.
  - Different layers exhibit different behavior. Recurrent layer representations become more generic in late training, but embedding layers become more specialized to their task late in training. However, embedding layers remain very generic throughout training, explaining the effectiveness of pretrained embeddings to initialize representations for other tasks.
- LSTMs Compose—and Learn—Bottom-up
  - We develop an extension of Contextual Decomposition called Decompositional Interdependence (DI). DI computes the level of nonlinear interactions there are between two words at a particular timestep, indicating the level to which they influence each other's "meaning" in context.
  - We test DI on an English LSTM LM, finding that when average DI is stratified by the word pair's sequential proximity (which is highly correlated with DI), higher DI indicates closer syntactic distance. This trend holds regardless of whether the words considered are closed or open POS classes.
  - We introduce the idea of a subtree acting naturally as *scaffolding* to build an interrelated meaning for a nearby item while predicting the next term in a sequence. In order to test the idea that known constituents act as scaffolding for longer-distance relations in LSTMs, we create a synthetic dataset with

long distance bracketing expressions surrounding constituents of varying familiarity. We find that poor generalization performance after training with familiar intervening spans can be attributed to the high interdependence between the constituent and the opening symbol of the bracketing expression. We conclude that LSTM learning is biased towards bottom up learning, using a known constituent as a scaffold to support new long distance rules.

- Sparsity Emerges Naturally in Neural Language Models
  - This work represents the first application in NLP of the Taxi-Euclidean norm for measuring soft sparsity.
  - We find that gradient concentration at the final RNN layer depends on a target word's POS class (open or closed), even after controlling for word frequency. Frequent target words from closed classes start out with highly concentrated gradients and soon stabilize, while frequent words from open classes continue to become more concentrated throughout training. We posit that this represents a transition from learning basic syntactic structure to learning general content words.
  - We find that the recurrent layers quickly learn to correlate sparsity with word frequency in their activations, but gradually the embedding layer surpasses the recurrent layers in this respect as the network converges.

Chapter 2

Background: Language Structure in Models

> "A slavish concern for the composition of words is the sign of a bankrupt intellect," roared the Humbug, waving his cane furiously.
>
> Norton Juster, *The Phantom Tollbooth*

A deep learning model operates by passing vector representations (also called **activations**) from one module **layer** to another during **inference** (i.e., when a text model makes a prediction). The model is not taught explicitly by humans to produce these representations. Instead, it is exposed to **training** samples, and performs gradient descent on the weights of its modules by **backpropagating** the gradient of the error of its predictions on these samples back to the first layer. Neither the inference nor training process is designed to be human-interpretable. Instead, these processes are surprisingly generic and can be used on many statistical models, and are highly efficient on modern hardware.

Despite widespread use and high accuracy, neural networks are therefore widely regarded as mysterious, "opaque and hard to interpret" (Elazar et al., 2020), leading to a "reputation of being black boxes" (Alain and Bengio, 2018). Their high performance does not translate to trust that they will handle edge cases that humans manage with ease, such as correctly identifying the subject of a verb and therefore choosing to inflect it correctly in a center embedded sentence like, "The best philosophers in the agora [is/are] with Socrates.." Much of the literature reviewed below carries the

implicit assumption that a model is more reliable if it operates internally more like a human[1].

Motivated by a desire for more trustworthy models as well as basic scientific curiosity, an abundance of recent work (Belinkov and Glass, 2019; Limisiewicz and Marecek, 2020) has focused on understanding the construction of a word representation during **inference** . In this chapter, we will survey a range of methods for understanding these models and the representations they use internally, which serve as key groundwork for the methods applied in our novel work. The chapter is ordered starting from methods that treat the model of interest as a black box and ending with methods that try to understand the intrinsic structure of the representations themselves. Some methods (Section 2.2) avoid direct access to the internal representations, probing with test data by measuring responses to particular inputs. Diagnostic Classifiers (Section 2.3) instead target intermediate representations by considering how they assist with auxiliary tasks. Finally, we explore methods (Section 2.4) that take full advantage of the access we have to the representational spaces that the model produces, measuring the intrinsic properties that align with human intuitions about how language data should be encoded. Before launching into these methods, we begin with a brief overview of LSTMs, the neural architecture we will focus on in this thesis (Section 2.1).

2.1 The LSTM Language Model

The novel work in this thesis focuses on experimental work on LSTM language models, which are **autoregressive** models, meaning they take a sequence as input and predict the next word (or character) at each point. These models have **recurrent** modules: they iterate over a sequence of arbitrary length, applying the same function to each word, with the output from the previous timestep included as a secondary input at the next timestep. In this case, the module uses both cell state c^t and hidden state h^t as inputs to the same function at the next timestep, along with the new input word x^t:

$$h^{t+1}, c^{t+1} = \text{LSTM}(h^t, c^t, x^t) \tag{2.1}$$

These recurrent functions are composed of a number of **gates**, each of which applies a nonlinear function to some combination of the input and hidden state. When we

[1] We note the exception of fairness research, where a model is often considered more reliable when it avoids human-like biases, in particular the tendency to learn prejudices like the assumption doctors are men because they frequently are so in the training corpus.

apply the by convention called the **forget gate** with activation f^t, the **input gate** with activation i^t, and the **output gate** with activation o^t, we produce the new **cell state** c^t and **hidden state** h^t, before moving on to the next input x^{t+1}. For each gate, we learn weights and bias, e.g., W_f and b_f respectively for the forget gate.

$$\begin{aligned} f^t &= \sigma(W_f x^t + V_f h^{t-1} + b_f) \\ i^t &= \sigma(W_i x^t + V_i h^{t-1} + b_i) \\ o^t &= \sigma(W_o x^t + V_o h^{t-1} + b_o) \\ \tilde{c}^t &= (W_{\tilde{c}} x^t + V_{\tilde{c}} h^{t-1} + b_{\tilde{c}}) \\ c^t &= f^t \cdot c^{t-1} + i^t \cdot \tilde{c}^t \\ h^t &= o^t \cdot \tanh(c^t) \end{aligned} \tag{2.2}$$

A multilayer LSTM would use h^t as input to the next LSTM module layer, and the last layer would use its output h^t to produce logits. These logits are used as inputs to a softmax function for a final distribution over target labels (in the case of language models, a word prediction).

$$\hat{x}^{t+1} = \text{Softmax}(W_L h^t) \tag{2.3}$$

Because the LSTM is trained by standard gradient descent methods and is not explicitly designed to be interpretable, it exhibits the same opacity that plagues most neural networks. The vector h^t, which represents the word in context as input for the next module, is the focus of particular attention, as is the memory cell c^t. In this chapter, we will consider a variety of methods to understand LSTM language models (and other text processing models) better.

2.2 Interrogation With Datasets

Some methods for understanding the behavior of neural networks are inspired by techniques in cognitive science of language (Bock, 1986; Miller and Chomsky, 1963; MacDonald et al., 1994; Futrell and Levy, 2017), where the near-total blackbox of the human brain permits access only to inputs and outputs[2]. In these cases, a stimulus is

[2]Although methods like fMRI analysis allow researchers to use some information about the intensity of regional activity in the brain, this data is far less than our perfect access to full information about each neuron's firing in real-time.

provided (for example, a syntactically ambiguous phrase) and the resulting behavior is read for an interpretation of the stimulus.

In order to understand the limitations of models in practice given natural language inputs, a variety of **challenge datasets** have emerged. Some are focused on fairness and the treatment of gender (Rudinger et al., 2018), but others have the sort of structural focus that targets the inductive bias of the training algorithm and architecture (Emelin et al., 2020; Linzen et al., 2016; Futrell et al., 2019). Some synthetic datasets take the form of augmented natural language sentences. These are deployed at test time in order to investigate whether the model has successfully acquired particular language rules, rather than using simplified heuristics. For example, Linzen et al. (2016) tested what kind of rule a network applies to match the inflection of a verb with its subject. They used examples with **distractor** noun phrases, as italicized in the sentence, "The best philosophers in the *agora* are with Socrates." A model that makes number inflection judgments purely based on proximity would assign higher probability to "is", which is predicted by the distractor noun ("agora", the most recent noun), rather than to "are" based on the actual subject ("philosophers").

2.2.1 Artificial Languages

In order to understand the inductive bias of neural networks that is imposed by the architecture and training of the model, we need to control the information yielded by the training data itself. This task is commonly accomplished through synthetic data. By manipulating training through synthetic data, we uncover the tendencies of model training under optimal and perturbed conditions. In these cases, the synthetic data needs to be introduced during training in order to yield informative results. Because we know how this training data was generated, we can test whether the model can emulate the generation process.

Synthetic datasets can mimic natural language to study biases that enable or inhibit grammatical rules. Ravfogel et al. (2019) generated synthetic versions of English with slightly different grammatical rules, training RNNs to predict agreement features for verbs. This task explicitly implements the decision that is implicit when an LM chooses between two verb inflections like "is" and "are". They found that overtly marking morphological case improved performance, as did using subject-verb-object word ordering (rather than artificially reordering the English sentences). Experiments like these reveal the ways in which the underlying inductive biases of a neural network

express themselves in language settings.

Mimicking natural language might also take the form of a template-generated subset of English. McCoy et al. (2020), for example, generated datasets that trained sequence-to-sequence networks to perform English question formation and verb reinflection, like the transformation from, "Diogenes does scandalize the Greeks," into, "Does Diogenes scandalize the Greeks?" However, they generated a training set that offered ambiguous evidence, so the model could either use the syntactic structure or simple sequential ordering. They then use test cases that required syntactic rules for reordering, such as "Philosophers who don't live in barrels do dislike Cynics". In the syntactic rule case, the model would move the main auxiliary of the sentence to the beginning ("Do philosophers who don't live in barrels dislike Cynics?"), whereas in the sequential case, it would move the first verb it encountered ("Don't philosophers who live in barrels do dislike Cynics?"). Because both latent tree structure and surface features were available, McCoy et al. (2020) could observe which rules each model favored. With real English generated from templates (Ravfogel et al., 2019), we can also test models pretrained in English on the biases acquired during training (Warstadt et al., 2020).

Bowman et al. (2015) and Hupkes et al. (2020) used artificial languages in experiments to test whether neural networks were capable of learning various patterns. Merrill et al. (2020) and Hewitt et al. (2020) instead proved analytically which artificial languages RNNs could learn in theory. (In a minor victory for RNN chauvinists, Hewitt et al. (2020) published a proof that RNNs could generate bounded-depth multi-symbol Dyck languages in subexponential memory, while simultaneously Bhattamishra et al. (2020) illustrated Transformers could not manage the same languages without custom positional encoding.) Weiss et al. (2018) measured the mismatch between rules an RNN learned empirically and the theoretical capacities of its model family.

We use simple artificial languages to test the natural behavior of LSTMs in Chapter 5, by testing how the model learns long-distance prediction rules in familiar or unfamiliar contexts. We *combine* this approach with inspection of the vector representations that the model uses internally, to investigate how these contexts are used.

2.3 Diagnostic Classifiers

While targeted datasets can measure the behavior of trained neural networks, they cannot reveal the internal logic of these models. Early efforts relied on **diagnostic classifiers** (DCs) to understand the construction of linguistic structure as representations are passed from one layer to the next. DCs are small peripheral models that use activations at various points in the forward pass as input representations to predict particular word labels (Belinkov et al., 2017; Hupkes and Zuidema, 2018; Adi et al., 2017; Conneau et al., 2018).

A typical example would see a simple linear model extract information like the length of the preceding sequence (Hupkes and Zuidema, 2018) or the part of speech of a particular word (Giulianelli et al., 2018), although nonlinear models have also been used (Belinkov et al., 2017). A simple DC, applied to hidden representation h^t, optimizes for accuracy of the predicted distribution on target linguistic property y^t, e.g., the word's part of speech:

$$\hat{y}^t = W_{\text{dc}} h^t + b_{\text{dc}} \tag{2.4}$$

If this linear classifier makes accurate predictions, the representation h^t is said to encode the property in question.

2.3.1 Criticisms of probing

Diagnostic classifiers and other methods of probing have met strong criticism in the NLP community. A frequent assumption researchers make when measuring the presence of a particular linguistic property with probes is the idea that the property is being used by the model and thus essential for model performance. Ravichander et al. (2021) questioned these fundamental assumptions, finding that DCs often detect linguistic properties that are not needed for a model's task. Furthermore, some information is generally preserved about the entire input sequence, e.g., a contextual word embedding may contain information as detailed as the word at a particular position in the context (Conneau et al., 2018). Therefore, if we permit arbitrary probe architecture, a DC could exhibit high accuracy on any solvable task (Pimentel et al., 2020b).

The ability of an arbitrary classifier to learn a function mapping from a contextual representation to a linguistic property does not tell us much. Does the vector contain information about the property (it would anyway; Pimentel et al., 2020b)? Does the model use information about that property (it still might not; Ravichander et al.,

2021)? We can't answer these questions based on the accuracy of a single probe. In fact, though some representations "probe" better than others, they may not be the ones we want. Popular contextual representations are often worse inputs for POS tagger probes than representations no one would use: randomly generated **unigram representation** vectors for each word, presented without incorporating any sentence context (Pimentel et al., 2020a; Zhang and Bowman, 2018)! For example, a vector from a randomized table entry for the word "philosophy" would be easier to POS tag with a diagnostic classifier, rather than a representation of the word "philosophy" that includes information about the entire sentence, "Diogenes hates philosophy homework".

Though the high performance of randomized unigram representations is bad news for the idea that a representation preferred by a probe is superior in general, it does tell us that learned word embeddings cluster together, making it harder to learn dictionary mappings from words to their most frequent part of speech. "Philosophy" is almost always a noun, and it's easier for the linear probe to learn a rule like that when each word is far from every other word. Even a learned unigram representations like word2vec Mikolov et al. (2013b) would learn similar representations for semantically related words like "philosophical", so the words might be mixed up while learning dictionary mappings (Pimentel et al., 2020a). This clustering behavior is a property of all used pretrained word embeddings. Therefore, we should be more concerned about what a probe reveals about the *intrinsic structure* (Torroba Hennigen et al., 2020) of a representation space, rather than whether a vector encodes linguistic information in some general way. Section 2.4 will introduce some probes designed to reflect intrinsic structure instead.

Constraining Models While this thesis focuses on the intrinsic structure of contextual word and sentence representations, rather than on probing methods, it is worth noting that many interpretability researchers have criticized classical probing methods based on issues like the preference to learn unigram dictionary mappings, and the community has responded by limiting the complexity of the probes. First, the vectors being probed contain trace information about the entire context (Conneau et al., 2018), rendering all linguistic information available to a sufficiently complex probe (Pimentel et al., 2020b). Some literature considered the linearity of the probe to be a significant enough cap on complexity (Liu et al., 2019a; Alain and Bengio, 2018), but not all representations are "expecting" to be processed by a linear classifier, rather than a softmax function or other deeper layers; these classifiers often have higher perfor-

mance on completely random unigram representations or other baselines (Hewitt and Liang, 2019); and there is little discussion of the broader implications when we *do* discover a particular property is linearly encoded. More recent work has developed explicit constraints on model capacity (Hall Maudslay et al., 2020; Hewitt and Liang, 2019). Pimentel et al. (2020a)[3] went further in measuring a property's encoding according to the *trade-off* between complexity and accuracy, rather than just measuring the accuracy of a single trained probe with a particular complexity.

However, methods based on extending or constraining DCs do not address our primary criticism of these probes as a method of model interpretation, which is that the classification accuracy of a particular probe model family does not give us any *interpretable* model of the structure of the representation space. While methods based on DCs remain popular thanks to their generality across both architectures and linguistic properties, it is not clear what information we glean about linguistic structure from these metrics. To gain insight into how the representation space behaves intrinsically, we will instead resort to structural probes.

2.4 Structural Probes

What does it mean to a human that a logistic regression of one layer is capable of extracting part of speech information from a particular representation? In response to the simplistic interrogation of models through classifiers (Section 2.3), we may ask whether these interpretation methods are themselves interpretable, and interpret model structure through **intrinsic** or **structural probes** (Torroba Hennigen et al., 2020).

Fortunately, work has emerged that inspects contextual representations by examining their geometry (Section 2.4.1) or comparing them to other representational spaces (Section 2.4.2). Other researchers fell back on techniques that identified which words in a sentence were most important for prediction (Section 2.4.3), though new attentional models suggested more intuitive and faithful ways of measuring word importance (Section 2.4.3.1). Approaches like these, which we now address, form our understanding of the intrinsic properties of vector representations.

[3] Co-first-authored by Naomi Saphra while writing this thesis.

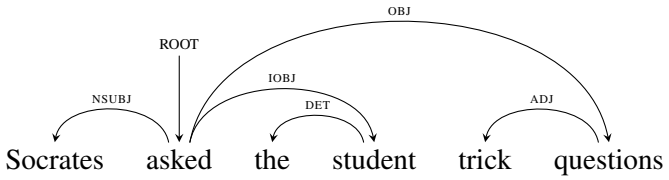


Figure 2.1: A dependency parsed sentence. The syntactic distance between "Socrates" and "trick" is 3 because there are three edges to traverse between them.

2.4.1 Vector Geometry

One alternative to the DC approach of interpreting vector representations is to consider the geometric relations between those representations and compare them to known language properties. For years, word embeddings have been evaluated based on their clustering of similar words and even on vector arithmetic often said to correspond to analogical reasoning (Mikolov et al., 2013a; Pennington et al., 2014; Saphra and Lopez, 2016).

Although it was common to use geometric and subspace methods to view unigram word embeddings (Mimno and Thompson, 2017; Mu and Viswanath, 2018), contextual word vectors presented an opportunity to understand representations in terms of their underlying linguistic properties within a sentence. Each word now corresponded to infinite possible representations based on its context (Ethayarajh, 2019), so instead of viewing words in terms of their relations in an abstract dictionary, we could analyze relations between specific occurrences of words.

Hewitt and Manning (2019) attempted to align arbitrary vector representations with the notion of syntax by learning a projection from the representational space produced by the model onto a space where distances between words resembled **syntactic distance**, the number of edges between words on a dependency tree (Figure 2.1). They found that syntactic distance was similar to the square of the Euclidean distance after a learned linear projection. Reif et al. (2019) suggested a possible explanation for why the *squared* distance was needed: trees cannot isometrically map onto a Euclidean space.

Consider "Socrates reads Diogenes polemicals". For an embedding in which $d(x,y)$ reflects the parse distance between x and y in the sentence context, we should see:

$$d(\text{Socrates}, \text{Diogenes}) \approx d(\text{Socrates}, \text{reads}) + d(\text{reads}, \text{Diogenes})$$

which in a Euclidean space would mean that these three words are embedded along a

straight line. In the same embedding space, we should find that

$$d(\text{Socrates}, \text{polemicals}) = d(\text{Socrates}, \text{reads}) + d(\text{reads}, \text{polemicals})$$

and again these three should be collinear—but this is only possible if "polemicals" has the same embedding as "students". In order to avoid this collision, the square term embeds each word at the nodes of a hypercube in a *Pythagorean* space, where the Euclidean collinearity axiom no longer applies.

Geometric analyses like these are appealing because they are intuitive, and seem to explain the underlying mechanisms of models. But they require commitment to a specific well-understood geometric space like the Pythagorean embedding, interpretations that do not always underlie highly complex, overparameterized models. More flexible structure analyses usually fall into the general category of **similarity analysis**.

2.4.2 Similarity Analysis

Hewitt and Manning (2019), described above, took the view that a learned representation had syntactic structure if its distances (after a learned projection) resembled syntactic distances. In general, we might like to say that the parse tree representation is *similar* to the learned representation, and therefore have some *shared structure*.

This philosophy has inspired a number of methods for understanding the structure of a representation space. To perform a similarity analysis, we take two ways of viewing a word and use some **similarity metric** to measure how much underlying structure those representations share. We might be comparing a learned representation to an explicitly structured representation like the parse trees in Hewitt and Manning (2019). More often we are comparing the subspaces or manifolds covered by two different learned representations. Let's look at some common methods.

2.4.2.1 Canonical Correlation Analysis (CCA)

First, we introduce CCA, a method for *computing the similarity of two different subspaces* while *removing information that's not shared between them*. Let us consider the case in which we want to compute the similarity of two different vectors $a^t \in \mathbb{R}^{d_1}$ and $b^t \in \mathbb{R}^{d_2}$, which represent the same word x^t. For each data point sampled—in our case a word, with two different vector representation—each vector forms a row of a matrix: $A \in \mathbb{R}^{n \times d_1}, B \in \mathbb{R}^{n \times d_2}$, where each row pair a^t, b^t represents the same data differently. Using the classic matrix factorization method of **Singular Value Decomposition**

(SVD), **Canonical Correlation Analysis** (CCA) learns linear projections from matrix representations. In order to find these projections, we identify vectors $u \in \mathbb{R}^{d_1}, v \in \mathbb{R}^{d_2}$ such that $u^T a^t$ and $v^T b^t$ are maximally correlated over all samples i. This correlation is then used as a similarity metric across the matrix representations.

The objective of CCA, then, is to maximize the **Pearson correlation** (Formula 2.5) of the projections.

$$\rho(x,y) = \frac{\text{Cov}(x,y)}{\sqrt{\text{Var(x)}}\sqrt{\text{Var(y)}}} \tag{2.5}$$

Recall the definitions of these statistical properties:

$$\text{Var(X)} = \mathbb{E}[X - \mathbb{E}[X]]^2 \tag{2.6}$$

$$\text{Cov}(X,Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \tag{2.7}$$

Then, if we have centered A and B, we can simplify the objective as:

$$\rho(u^T a, v^T b) = \frac{\mathbb{E}[(u^T a)(v^T b)]}{\sqrt{\mathbb{E}[u^T a]}\sqrt{\mathbb{E}[v^T b]}} \tag{2.8}$$

$$= \frac{\mathbb{E}[(u^T a b^T v)]}{\sqrt{\mathbb{E}[u^T a a^T u]}\sqrt{\mathbb{E}[v^T b b^T v]}} \tag{2.9}$$

$$= \frac{u^T K_{AB} v}{\sqrt{u^T K_{AA} u}\sqrt{v^T K_{BB} v}} \tag{2.10}$$

with K_{AA} and K_{AB} the covariance and cross-covariance matrices, respectively. We can constrain the denominators to 1, as rescaling either u or v by any constant will not affect the objective.

Therefore, we have turned the objective into a convex optimization problem

$$\begin{aligned} \max_{u \in \mathbb{R}^{d_1}, v \in \mathbb{R}^{d_2}} \quad & u^T K_{AB} v \\ \text{s.t.} \quad & u^T K_{AA} u = 1 \\ & v^T K_{BB} v = 1 \end{aligned} \tag{2.11}$$

This constrained optimization problem forms a generalized eigenvector problem, meaning it can be solved repeatedly by choosing the top eigenvectors. Thus we can choose an arbitrary dimension no larger than the rank of the lower-rank matrix between X, Y for the space data is projected onto. This makes CCA a rank-reduction method, though one which simultaneously reduces the rank of two different views of the data.

2.4.2.1.1 Variations on CCA CCA has been expanded in a number of ways. **Singular Value Canonical Correlation Analysis** (SVCCA; Raghu et al., 2017) uses the representation matrices produced by each layer of a model as inputs to CCA, but adds an extra SVD step before CCA to denoise the representations before finding their similarity. This rank reduction is performed by factorizing each representation matrix and then setting the smallest singular values to 0, constructing the new representation matrices as the product of the modified factors. SVCCA is a key component of the probing method in Chapter 4. **Projection Weighted CCA** (PWCCA; Morcos et al., 2018a) and **Centered Kernel Alignment** (CKA; Kornblith et al., 2019) propose other methods of adapting CCA to the noisy representations produced by neural networks. In addition to the use of SVCCA in Chapter 4, these correlational subspace methods continue to see use in interpretability work (Voita et al., 2019a; Wu et al., 2020a; Hao et al., 2020).

2.4.2.2 Representational Similarity Analysis (RSA)

RSA is another common way of evaluating the similarity of two views of the same data. NLP researchers borrowed this method from systems neuroscience (Kriegeskorte et al., 2008) in order to analyze neural networks (Chrupala and Alishahi, 2019; Chrupala, 2019; Lepori and McCoy, 2020). In this method, the two views are in the form of two **kernels**, or similarity functions. By using kernels, we don't need to have to use vector representations for both views of the data. For example, you may have on one side a *tree kernel* that considers proximity on a syntax (as the edge distance above from Hewitt and Liang (2019)). As the other kernel view, you use cosine similarity or some other vector operation comparing the representations produced by a neural network. RSA similiarity is then the **Spearman correlation** between these kernels, which is the Pearson correlation (Formula 2.5) between the rank of distances in the kernels. Using rank in this way has the advantage of eliminating scaling as a factor; e.g., if we compared similarity function $k(x)$ to $\log(k(x))$, Spearman would recognize them as perfectly correlated.

2.4.2.3 Comparison

Wu et al. (2020a) performed an analysis of different measurements of similarity, revealing differences between both interpretation methods and the models they were applied to. They confirmed that models with different architectures (RNN- and Transformer-

based models), trained on the same task, produced similar representations according to similarity metrics that considered dimensions that were distributed across neurons (SVCCA, PWCCA, and CKA). In contrast, they found that metrics based on the directions of individual neurons did not detect these similarities, supporting the claim in Morcos et al. (2018b) that features are encoded in directions that span many neurons[4].

Implicit in Chapter 4 is the assumption that models trained on the same tasks should produce similar representations. Wu et al. (2020a) not only confirmed this expectation with respect to the SVCCA metric, but also showed that models within the same family are more similar. This last finding further validated similarity analysis as a way of investigating whether particular modules play similar roles within their respective networks.

2.4.3 Word Importance

Early efforts to assign weights to words and phrases in contextual models borrowed saliency methods from computer vision (Simonyan et al., 2014)[5]. These techniques base importance on the magnitude of activation vectors (Karpathy et al., 2015) and gradients (Li et al., 2015). In language, it is also possible to directly test importance by removing words (Arras et al., 2016; Li et al., 2016), but such methods neglect the sequential interactions and instead treat sentences inappropriately as bags-of-words, as argued by Feng et al. (2018).

2.4.3.1 Attention distributions

Attention modules reweight different features, often words, before adding together the vectors associated with each feature. One of its most common applications is in sequence-to-sequence (**seq2seq**) tasks, where the **Transformer** model dominates leaderboards by relying entirely on attention modules (Vaswani et al., 2017). The Transformer architecture applied to a sequence of length n includes three learned components: a query $\mathbf{Q} \in \mathbb{R}^m$, a key $\mathbf{K} \in \mathbb{R}^n$, and a value $\mathbf{V} \in \mathbb{R}^n$. The query is a compressed representation of previous output, and the key-value pairs encode the set of inputs. The attention itself is then computed as:

[4]However, Wu et al. (2020a) did find that individual neurons were similar in models from the same **model family**; for example, LSTMs had neuron-level representations that were similar to other LSTMs.

[5]New analytic work in machine learning tends to appear in computer vision first, which is why most of the work in Chapter 3 focuses on that domain.

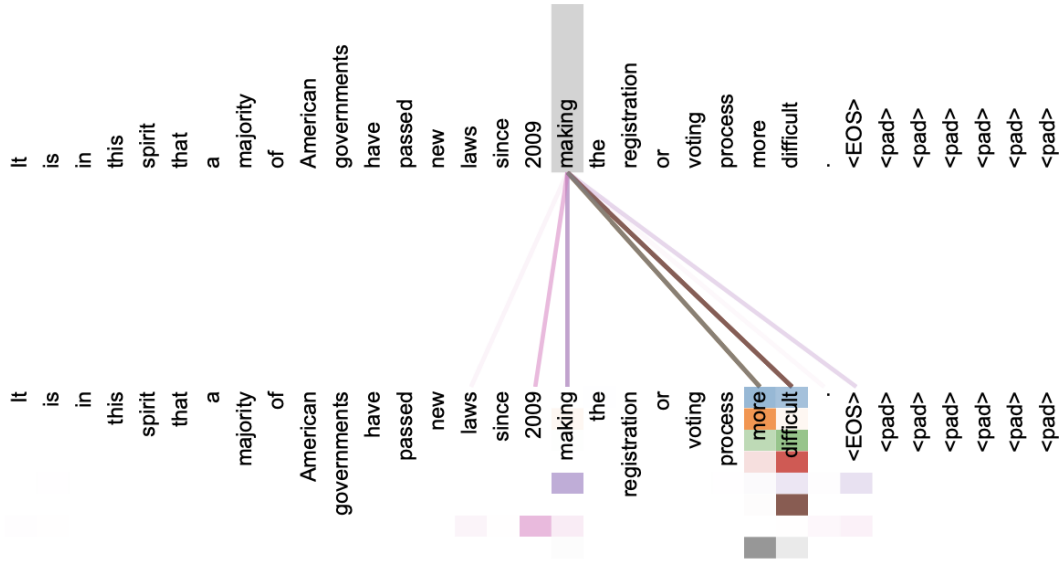


Figure 2.2: Multi-head attention (with different colors representing different heads) in predicting "making", from Vaswani et al. (2017). Strong weights are placed on "more difficult", a phrase directly syntactically linked to "making".

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}}\right)\mathbf{V} \tag{2.12}$$

The weight distribution produced by the softmax function has an implicit interpretation as "relevance", strengthening connections between words in a way that draws tempting parallels to latent graph structures in language, like syntactic dependency (Clark et al., 2019; Voita et al., 2019b; Htut et al., 2019) or constituency (Marecek and Rosa, 2019). This relevance interpretation is seen clearly in Figure 2.2, where a close syntactic link in the stereotyped expression "make *noun* more *adjective*" is highlighted by the attention distribution. Such methods are convenient but usually limited to attention-based models[6].

In settings where we have only one attention distribution when labeling each word, we can ask whether the attention distribution points to, e.g., a parent word or coreferent, as in the *making-more* link in Figure 2.2. In architectures with **multi-head** self-attention, multiple attention distributions operate at each layer, so we can no longer point to a single dominant word in a single distribution (Limisiewicz and Marecek, 2020). Instead, a common way to measure the syntacticity of the attention distributions is to

[6]Attention can sometimes be added as a peripheral inactive module for interpretation purposes (Goddard et al., 2018).

count the performance of the "best" head for a particular relation or instance of the relation (Voita et al., 2019b; Clark et al., 2019). This strategy considers a particular head to be best for a relation like subject-verb, that focuses on a different head in calculating syntacticity when a different relation appears, such as object-verb.

In general, the topmost encoder layers of machine translation (MT) models and the middle layers in Transformer-based LMs like BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) align with syntactic intuition (Limisiewicz and Marecek, 2020).

2.4.3.1.1 Is attention interpretable? Debate rages over whether attention can be interpreted as word importance in this way (Jain and Wallace, 2019; Serrano and Smith, 2019; Wiegreffe and Pinter, 2019; Vashishth et al., 2019; Brunner et al., 2020), not least because of their limited capabilities on long sequences (Hahn, 2020; Brunner et al., 2020) and the availability of model-agnostic alternatives like saliency (Bastings and Filippova, 2020). Regardless of their faithfulness and validity, attention-based analyses of structure are simple to implement and often align with intuition in their results. Even in theory, key-value pairs may mutually amplify each other during training (Lu et al., 2021), so we can generally gain some intuition about the internal workings of attention models by observing their weight. Attention-based interpretation methods therefore continue to be in common use today (Belinkov and Glass, 2019; Rogers et al., 2020), and observations of attention distributions during training could help extend much of the work in this thesis to modern attentional models.

2.4.4 Contextual Decomposition

If we want to consider not just the importance of each word in a sequence, but the causal contribution that entire sets of words make towards a model's output vector, we can use **Contextual Decomposition** (CD), proposed by Murdoch et al. (2018). CD is a way of approximating these influences in an LSTM without adding an attention mechanism or training an additional model (as in DCs).

In the language of CD, we consider the hidden and output vectors produced by an LSTM module to be the sum of **relevant** and **irrelevant** parts, the contributions respectively of the **in-focus** and **out-of-focus** words in a sequence. For example, we might decompose the hidden layer into contributions of the relevant word set β and the irrelevant word set $\bar{\beta}$, producing:

$$h = h^t_\beta + h^t_{\bar{\beta}} \tag{2.13}$$

This is not a trivial decomposition, because the LSTM module applies nonlinearities at each gate, the sigmoid and tanh functions, as shown in Section 2.1. CD therefore finds an approximate decomposition of the hidden state by linearizing each gate operation.

For example, Murdoch et al. (2018) use a linearized approximation L_σ for σ (and similarly a linearized approximation $L_{\tanh}$ for tanh) such that for arbitrary input $\sum_{j=1}^{N} y_j$:

$$\sigma\left(\sum_{j=1}^{N} y_j\right) = \sum_{j=1}^{N} L_\sigma(y_j) \tag{2.14}$$

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:

$$\begin{aligned} i^t &= \sigma(W_i x^t + V_t h^{t-1} + b_i) \\ &\approx L_\sigma(W_i x^t) + L_\sigma(V_t h^{t-1}) + L_\sigma(b_i) \end{aligned} \tag{2.15}$$

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 2.14 to $\sigma(y_1 + y_2)$ for a linear approximation of the isolated effect of the summant y_1:

$$L_\sigma(y_1) = \frac{1}{2}[(\sigma(y_1) - \sigma(0)) + (\sigma(y_2 + y_1) - \sigma(y_1))] \tag{2.16}$$

With this function, we can take a hidden state from the previous timestep, decomposed as $h^{t-1} \approx h^{t-1}_\beta + h^{t-1}_{\bar{\beta}}$ 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:

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

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.

In order to isolate h^t_β from the context, the irrelevant component contains all nonlinear interactions between in-focus and out-of-focus words. We extend CD to analyze these interactions between words in Chapter 5, leading to a new perspective on how an LSTM constructs meaning over the course of training.

Although CD is particular to the LSTM architecture, the principle behind it is that of the Shapley decomposition, which is general across architectures. Therefore, this generalization of word importance has parallels in other architectures. Shapley values have been used to understand the contribution of entire sets of words over an entire activation vector in many environments Lundberg and Lee (2017); Chen et al. (2019); Ghorbani and Zou (2020); Zhang and Nie (2020). The existence of such work suggests possible generalizations of any research based on CD to other architectures.

2.5 The Perils of Creationism

For centuries, Europeans agreed that the presence of a cuckoo egg was a great honor to a nesting bird, as it granted an opportunity to exhibit Christian hospitality. The devout bird enthusiastically fed her holy guest, even more so than she would her own (evicted) chicks (Davies, 2015). In 1859, Charles Darwin's studies of another occasional brood parasite, finches, called into question any rosy, cooperative view of bird behavior (Darwin, 1859). Without considering the evolution of the cuckoo's role, it would have been difficult to recognize the nesting bird not as a gracious host to the cuckoo chick, but as an unfortunate dupe.

Whether looking at parasitic brooding behavior or at the inner representations of a neural network, if we do not consider how a system develops, it is difficult to distinguish a pleasing story from a useful analysis. In NLP, how can we know if a pattern emerges as informative structure which is used by the model? Apparent syntactic patterns may well be vestigial effects from strategies early in training,[7] or even side effects of training, input structure encodings with no bearing on the final predictions. We consider similarity (Chapter 4) and sparsity (Chapter 6) not at one checkpoint, but throughout training. We even illustrate how an LSTM's training strategy makes it well-suited to

[7]For one origin story, see the Information Bottleneck Hypothesis, Section 3.1.2.

hierarchical structured data (Chapter 5). These analyses lend a dimension to our understanding that is missing from the "creationist" view of a single checkpoint, manifested without history or evolution.

Chapter 3

Background: Training Dynamics

> The most important reason for going from one place to another is to see what's in between, and they took great pleasure in doing just that. Then one day someone discovered that if you walked as fast as possible and looked at nothing but your shoes you would arrive at your destination much more quickly.
>
> Norton Juster, *The Phantom Tollbooth*

The work in this thesis focuses on interpreting the development of linguistic structure over the course of training a language model. It therefore appears at the intersection of two fields: the first being language model interpretability (Chapter 2), the second **training dynamics**. The field of training dynamics focuses on *understanding how the strategies used in training neural networks actually work in practice*.

One example of training dynamics research is the thread of work which describes how a model becomes more complex throughout training. This complexity may be measured according to how easy it is to find adversarial examples; early in training, an image classifier can be easily tricked with slightly perturbed data (Arpit et al., 2017). Another way of measuring the same phenomenon is by comparing the classification accuracy on various samples between a state-of-the-art image classifier and a shallow model; the deep network behaves remarkably like classical shallow models earlier in training (Mangalam and Prabhu, 2019).

So now we know that, by observing the networks throughout training, we can conclude that deep networks evolve from modeling shallow patterns. An understanding of the training process requires a mix of empirical investigations like this as well as more

theoretical work. *This chapter will focus on the research in training dynamics that we draw on, either in methodology or in formulating hypotheses about language model behavior during training.*

The difficulty, as well as the opportunity, of working with training dynamics in NLP is that the extensive research done on understanding and modeling training dynamics is almost exclusively focused on the computer vision domain. Computer vision uses continuous data as input, and currently tends to investigate the dynamics of feedforward networks, convolutional networks, and residual networks. We have little information on the dynamics of RNNs and Transformers, which are common in NLP—or on how dynamics will differ when the input data takes the form of simplistic discrete sequences with latent tree structures, let alone the fuzzy, complex structure of actual language.

Before considering this research gap, in this chapter we will overview some of the existing understanding of training dynamics in neural networks. The work we discuss is often based on experimental research in computer vision, or on synthetic tasks developed by computer vision researchers. Many of the experiments are based on perturbations or measurements with no obvious parallel in NLP.

In language, meanwhile, an entire field of linguistics is dedicated to observing ideal learners in action: human children. Developmental linguistics has a long history of investigating language learning through probing the emergence of structure, but these patterns may not be observed in neural networks. For example, a classic pattern observed over the time course of human learning is the "U-shaped curve" (Plunkett and Marchman, 1991). In early childhood, a learner memorizes verb forms, both regular verbs like "walk" inflecting as "walked" and irregular verbs like "run" inflecting as "ran". Later on, the child learns a general inflection rule, so instead of using the memorized forms, they are likely to say words like "runned". In order to speak grammatically like an adult, the child has to memorize the exceptions to these rules again. Kirov and Cotterell (2018) claimed to observe classic U-shaped curves in a network learning irregular verbs, but Corkery et al. (2019) found it to be an anomaly among model runs. It is rare to see such a debate on whether modern neural networks use humanlike strategies in learning language—or any recent investigation into the learning strategies of language technologies. We see this gap as an opportunity, a space flourishing with questions about how the structure and representation of data influence model training.

Whether or not an artificial neural network experiences the memorization to overgeneralization to correction transitions indicated by a human learner's U-shaped curve, there are many indications that it undergoes its own **phase transitions** during training (Section 3.1). Furthermore, theoretical work on the trajectories followed by gradients can shed light on how underlying data ontologies can make one architecture more suitable than another (Section 3.2). And where theory fails, we can turn to a variety of empirical methods for understanding the training process (Section 3.3). The existing work in training dynamics provides both questions about how language models learn, and methods for finding solutions.

3.1 Learning in phases

Practitioners in machine learning have found that switching optimization strategies early in training can be more effective or efficient; understanding these shifts, as well as other "natural" changes in a model's learning strategy, is a fundamental problem in training dynamics. For example, BERT's training regime switches one hyperparameter, batch size, dramatically midway (Devlin et al., 2019). Although smaller incremental changes in step size have long been in practice[1], the advantage of large modification of other hyperparameters during training (Lee et al., 2020; Popel and Bojar, 2018; Bogoychev et al., 2018) may be linked to recent evidence that training naturally occurs in multiple phases (Jastrzebski et al., 2020; Shwartz-Ziv and Tishby, 2017; Kaplan et al., 2020). A phase-based analysis of learning takes the view that the early stages of learning have distinct behavior and purpose compared to later stages. These phase transitions emerge in a number of settings and using various methods of modeling the learning process.

Phase transitions mark changes in how model generalization or strategies respond to how much training data a language model has consumed. Kaplan et al. (2020) studied both LSTM and Transformer language models and found that loss scales as a power law in response to dataset size and amount of total training time, with plateaus at the beginning and end of training, shown in Figure 3.1. At the end of training, this plateau in performance occurrs significantly before model convergence, indicating that training until convergence is inefficient if the only goal is language modeling. However, modern language model training is often aimed at pretraining a model for a different task,

[1] More rarely, other hyperparameters can receive similar treatment of incremental changes, as in curriculum dropout (Morerio et al., 2017).

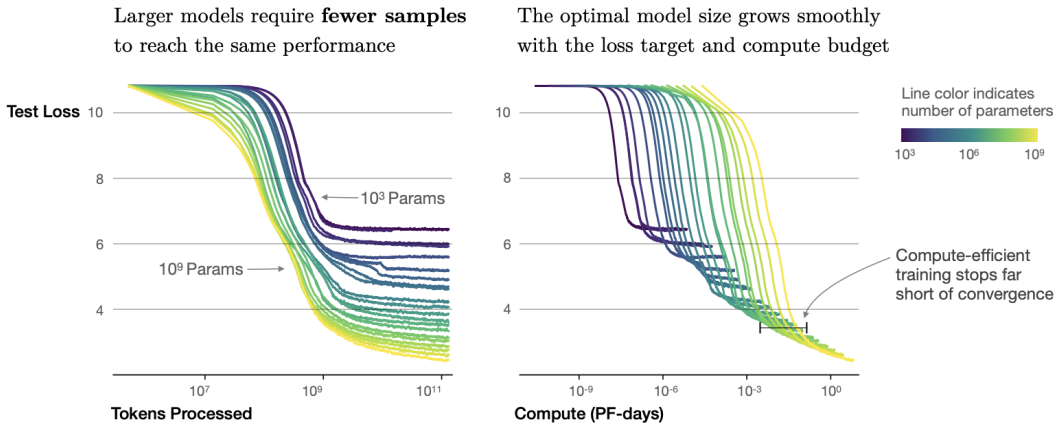


Figure 3.1: Various sizes of LM each show that loss scales as a power law in terms of both training time and data size, though with a lag at the beginning and saturation at the end. Reprinted from Kaplan et al. (2020).

and training beyond apparent convergence seems to improve transfer performance, a source of some of the improvement RoBERTa (Liu et al., 2019b), one large Transformer language model, saw over the previous state-of-the-art, BERT (Devlin et al., 2019). Phase transitions appear across many models and domains; Jastrzebski et al. (2020) identified points in training, conditioned on early stage step size, after which SGD acts as an implicit regularizer for both gradient curvature and data noise. They identified these **break-even points** in vision, medical, and NLP tasks.

Linguistic structure emerges slowly Kaplan et al. (2020) found that test error on a language modeling objective was strongly correlated with training error throughout training. However, Warstadt et al. (2020) found a threshold in training data size that allows RoBERTa to prefer underlying linguistic structure over surface features when fine-tuned on various tasks. That is, the model continues to encode linguistic properties in useful ways late in training. As one example of a linguistic task, they pose a morphological binary classification task, asking whether the main verb ends in "-ing". Efficiently solving this problem with word-level embeddings requires clustering on the basis of inflection, because the model does not have direct access to the character sequence that encodes "-ing" as a surface feature. However, some of the training examples could also be classified by asking a word-level, and therefore surface-level, question: Does the sentence contain the word "the", as in, "The great philosopher is questioning Diogenes"? This creates an *ambiguous* portion of the training set. By monitoring how much *unambiguous* training examples the model needs in order to

Figure 3.2: Reprinted from Hirsch and Spinelli (1971), a kitten wearing goggles to limit their vision. As an adult cat, their vision will remain adapted to the obstructed view offered by the goggles.

learn the underlying morphological property, instead of the surface property, Warstadt et al. (2020) found that RoBERTa began to prefer such linguistic properties only after consuming large amounts of data, between 1B and 30B words. This transition to a preference for linguistic properties may constitute one phase transition particular to language models.

3.1.1 Critical Learning Periods

Other phase transitions have been observed only in computer vision. Achille et al. (2019) discovered a substantial limitation of neural networks when confronted with perturbed training data: They could not recover from an early training stage if they blurred the original input images. Instead, the models continued to make accurate predictions on blurred images and failing at the original sharper images, even after converging while training on the original image corpus. They connected this tendency to animal critical learning periods, the phenomenon that limits brain plasticity when responding to a change in environment after early development. Classic work in understanding animal critical learning periods relies on manipulating the senses of growing cats (Wiesel and Hubel, 1963; Hirsch and Spinelli, 1971). Kittens, raised from infancy with goggles that limit their view (Figure 3.2), adapt to this environment and never ac-

quire normal visual capacity. Manipulations like this are effective only during early development, so an adult cat will not lose normal vision by wearing the goggles.

Particular critical learning periods are yet undiscovered in NLP models. NLP tends to focus on the related question of what properties are useful or transferred from pre-trained models before fine-tuning (Papadimitriou and Jurafsky, 2020; Warstadt et al., 2020). For instance, it seems the very existence of matching parentheses—even without consistently nesting them—is useful information for an LSTM to base an English LM on (Papadimitriou and Jurafsky, 2020), telling us that the LSTM module is used to keep track of individual pairs of associated words.

3.1.2 Information Bottleneck Hypothesis

(Achille et al., 2019) found that critical learning periods (Section 3.1.1) aligned during training with a transition from input memorization to input forgetting. This pattern of phase transition during training was first documented by Tishby and Zaslavsky (2015). They based their work on concepts from information theory, and in particular on **mutual information**. Mutual information is a measurement of the dependence between two variables A and B, expressed as:

$$\mathrm{I}(A;B) = D_{KL}[p(A,B)\|p(A)p(B)] \tag{3.1}$$

This is the **KL-divergence** between the joint distribution of A and B and their independent distributions.

Their proposal was not unlike the U-shaped curve: in the early stages of training (the **empirical risk minimization (ERM) stage**), a neural network naturally memorizes its inputs because its tendency is to *maximize* mutual information between the *input* and the *internal representation*. It then reaches the **representation compression** or **forgetting stage**, wherein it *minimizes* mutual information between the *input* and the *internal representation*–while maintaining as much information as possible about the *output*. Like in a human learner's U-shaped curve, the network experiences a memorization phase followed by learning a simpler, and therefore more general, shortcut. Tishby and Zaslavsky (2015) further propose that fully optimized neural nets would approach the **Information Bottleneck (IB)** bound (Tishby et al., 2000), where the representation compression stage reaches the optimum selecting intermediate representations T between the input X and output Y such that

$$\hat{T} = \min_{T} I(X;T) \text{ s.t. } I(T;Y) = \eta \tag{3.2}$$

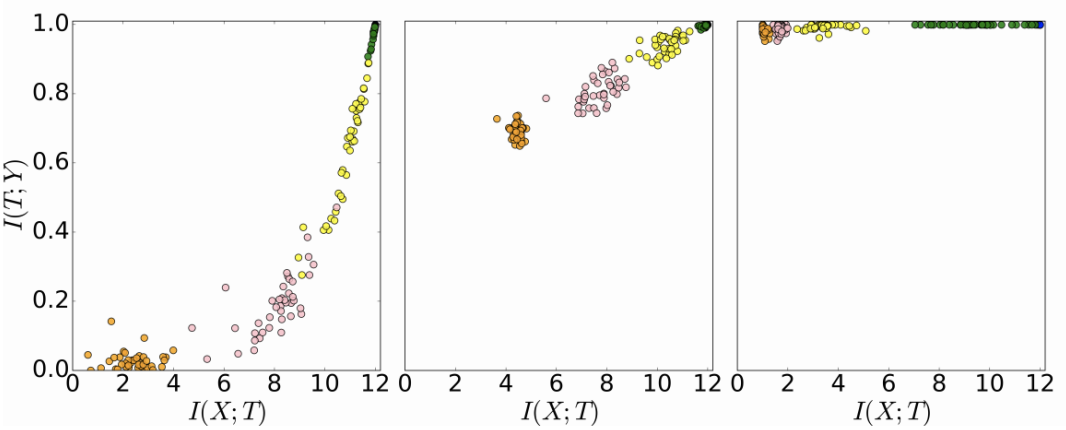


Figure 3.3: Reprinted from Shwartz-Ziv and Tishby (2017), an information plane analysis of representations T of a neural network case study. From right to left, plots are of mutual information at initialization, 400 epochs, and 9000 epochs. Colors specify the layer. During training, each layer first increases input information $I(X;T)$, then reduces it while maintaining maximum $I(Y;T)$ information retained by the layer about the output label.

minimizing information flow from X to T while maintaining a constant information η passed from T to Y.

Shwartz-Ziv and Tishby (2017) illustrated the two optimization stages on the **information plane** (Figure 3.3). The information plane covers two dimensions, $I(X;T)$ and $I(T;Y)$. During SGD, they found that $I(X;T^{(i)})$ and $I(T^{(i)};Y)$ both increased at layer i until $I(T^{(i)};Y)$ reached its maximum, at which point $I(X;T^{(i)})$ began to decrease.

While Shwartz-Ziv and Tishby (2017) used a synthetic task and toy model to demonstrate the information bottleneck hypothesis of training stages, Voita et al. (2019a) applied the information bottleneck principle to the layer-wise development of representations in a Transformer Masked Language Model. Using variations on CCA (see Section 3.3.2), they illustrated how lower layers have high similarity with both input and output data, while similarity to the input is reduced gradually as higher representations grow closer to the output layer.

3.1.2.1 Controversy

Shwartz-Ziv and Tishby (2017) selected a simplistic setting with a toy task in order to simplify the process of deriving mutual information. Theoretically, mutual information

is a claim about what information is transmitted when considering all possible models to translate between input and output distributions, making it an intractable metric. Instead of measuring true MI, Shwartz-Ziv and Tishby (2017) discretized the units produced by each layer's *arctan* activation by binning them, thereby creating distributions which could be compared directly by KL-divergence.

Saxe et al. (2018) challenged the general applicability of the IB results. Their analyses and simulations found that this two-stage learning process, and its effects on generalization, were only found in the arctan activation setting. Simple linear networks and more common ReLU activations did not follow the same pattern. Saxe et al. (2018) found that the IB findings applied only when an activation function was saturated (i.e., exhibited asymptotic behavior) on *both* sides, unlike popular activations which saturated only at low values (e.g., ReLUs).

Rather than settling the case of whether the IB hypothesis applies in most learning settings, these results ignited a debate. Noshad et al. (2019) attributed the negative findings in the ReLU case to poor MI estimation methods, proposing the **EDGE** MI estimation method and using it to demonstrate a compression phase as predicted by Tishby and Zaslavsky (2015). Goldfeld et al. (2019) then validated the IB hypothesis in a variety of networks using another framework and estimator. While the information plane analysis Shwartz-Ziv and Tishby (2017) employed has found popularity in analysis of learning dynamics (Lanorte et al., 2014; Wickstrøm et al., 2019; Cheng et al., 2018), the ML community have not settled on the universality of the IB hypothesis.

In Chapters 4 and 6, we find shifts in behavior over the course of LM training that we view as phase transitions. These transitions can be read as further evidence of the IB hypothesis, as we comment on in the papers.

3.2 Gradient Trajectories and Underlying Semantics

Thus far, the rules of dynamics we have discussed are presented as largely model- and data-agnostic[2]. But we must not leave the impression that the field of training dynamics neglects these elements. In reality, the particulars of the data have a profound effect on learning. Swayamdipta et al. (2020) empirically demonstrated the influence of data by visually mapping individual training samples according to their impact on a model's

[2]While critical learning periods are *detected* through data manipulation, the nature of the data is not a central focus of Achille et al. (2019).

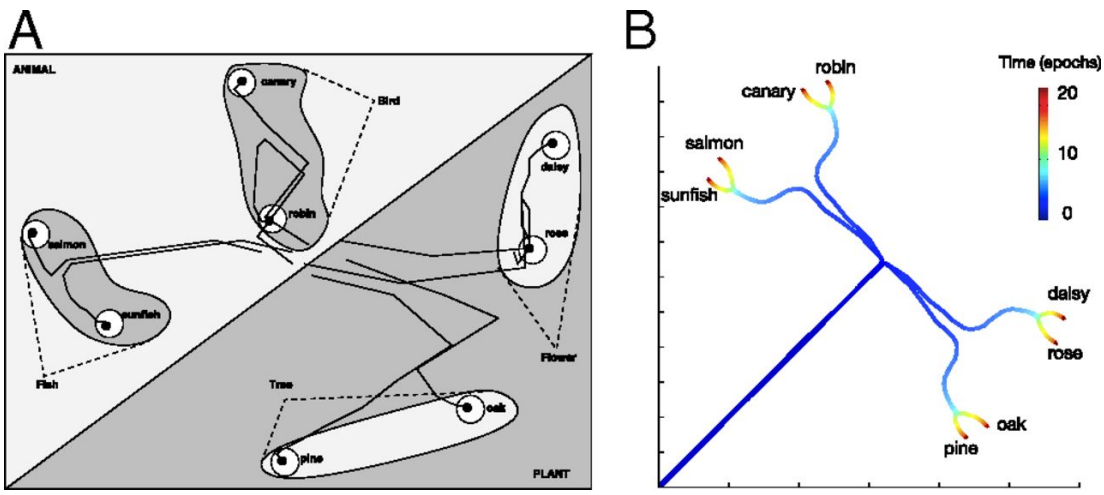


Figure 3.4: Reprinted from Saxe et al. (2019), (A) A multidimensional scaling (MDS) visualization of internal representations from a nonlinear NN, from Rogers and McClelland (2004) (B) Analytically derived MDS visualization of a tree-derived ontology according to Saxe et al. (2019).

correctness, *confidence*, and *variability*. They found that the most ambiguous (high-variability) samples were essential to out-of-domain generalization, and emphasized the importance of data curation.

We can view the effect of data structure more generally by considering idealized data sources. Saxe et al. (2019) proved and simulated the dynamics of a linear neural network trained to predict the features of types which are generated by a tree-based ontology. For example, the ontology might include a daisy as a type of flower, which is a category of plants, with properties like "has leaves" and "does not move". Penguins would be generated as a type of birds, which are a category of animals, but unusually among birds would have features like "does not fly" and "swims". Saxe et al. (2019) found that in a two layer neural network, animals and plants would be clustered altogether at the beginning, and gradually differentiated into their child categories like bird and tree, so that differentiation over the course of training followed the same structure as the tree ontology (see Figure 3.4).

Saxe et al. (2019) also proved that the addition of an extra layer to a one-layer linear network, turning it into in a two-layer linear network, would accelerate the trajectory of gradient descent in the dominant directions of the data. In this case, the dominant directions would identify boundaries close to the "root" of the ontological tree, like the plant/animal split in Figure 3.4. This modification does not affect which models

could hypothetically be learned, because a one-layer linear neural network covers the same function space as a two-layer linear neural network. However, it influences the weights learned by gradient descent. In practice, this tendency would decrease the noise from aberrational types like the penguin, emphasizing the clearest directions. This result about the impact of adding layers to a linear network was thematically extended by Arora et al. (2019), who found that performing matrix decomposition with deeper linear networks would provide natural regularization, because the trajectories in gradient descent increasingly neglected the least significant or "noisiest" direction as model depth grew.

Saxe et al. (2019) and Arora et al. (2019) highlight how understanding training dynamics helps identify the inductive biases that make one architecture well-adapted to a particular data source or underlying structure. This work is a significant inspiration for Chapter 5, where we investigate how hierarchically generated sequences—like idealized language generated at the leaves of a syntax tree—lend themselves to the LSTM model. By looking at the model during training, we see how its representations repeatedly merge, forming larger trees out of small constituents.

3.3 Exploring the Course of Training

We have discussed a variety of theories and results, but what if we want to empirically investigate the learning dynamics of an arbitrary neural network? In this case, we may need more practical tools.

3.3.1 Loss Visualization

When a human investigates a neural network's representations, we often project these representations onto a 2-, or possibly 3-dimensional image. A similar option is available when looking at training, where we can view the **loss landscape** of the model by modeling the change in the loss function outside of the immediate gradient. A simplistic method of deriving a loss landscape is by **Linear Interpolation** (Goodfellow et al., 2015), wherein one chooses two parameter settings as the vectors θ and θ' and then plots each point between them, sampling at some interval (Figure 3.5). The function plotted to visualize loss function L is $f(\alpha) = L((1-\alpha)\theta + \alpha\theta')$, sampling at various test points α. Among other problems, this technique is unreliable when modeling non-convexities.

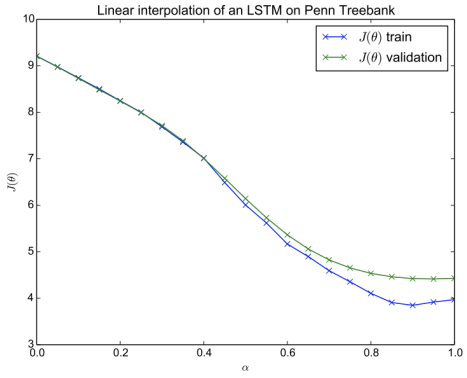


Figure 3.5: Linear interpolation of the loss on an LSTM trained on PTB, reprinted from Goodfellow et al. (2015).

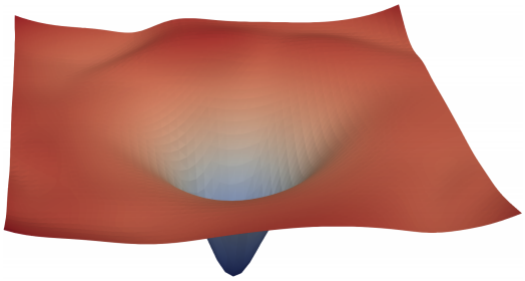


Figure 3.6: 3-dimensional loss landscape for ResNet-56 with skip connections, reprinted from Li et al. (2018).

To improve on linear interpolation, Li et al. (2018) and Goodfellow et al. (2015) use random directions to add test points to θ. In this case, they choose the random direction δ and plot $f(\alpha) = L(\theta + \alpha\delta)$. Unlike Linear Interpolation, random directions can generate 2- or 3-dimensional contour plots by selecting other random directions in addition to δ and making f a multivariate function. Using these methods, Li et al. (2018) showed that SGD optimization trajectories occupy an extremely low dimensional space, defined by large nearly-convex regions in the loss landscape (Figure 3.6). This near-convexity may characterize the "good" optima which generalize well.

Loss landscape methods visualize changes in error based on changes in parameter settings, but by varying parameters along only one or two dimensions, they cannot provide interpretable information about individual neurons or even units. To study the contributions of parameters on a granular level, we might turn to **Loss Change Allocation** (LCA; Section 6.1.1.1). On the other hand, if we wish to abstract away the role of each individual neuron entirely and observe only changes in the subspaces they

produce, we can analyze representational similarity, as follows.

3.3.2 Similarity Analysis

Raghu et al. (2017) originally devised SVCCA (Section 2.4.2) in order to observe the movements of neural networks during training. By visualizing the convergence of neural networks using linear projection methods, they illustrated neural networks learning naturally from the bottom up. They also measured when different classes were learned, by using CCA on the Discrete Fourier Transform between various layers and the logits; for example, ImageNet (Russakovsky et al., 2015) Resnet (He et al., 2016) distinguishes firetrucks from dogs before it learns to distinguish different dog breeds.

Morcos et al. (2018a) followed these results using PWCCA to analyze RNNs. Again, they observed that lower layers converged faster than upper layers in multilayer RNNs. They discovered the hidden state is highly variable even through this lens, indicating new words in a sequence are not just applying linear transformations to the state. This last observation is highly relevant to our use of approximations to measure the nonlinearity of transformations applied by various words in Chapter 5.

3.4 The Perils of Mono-Domainism

We know little about how language models learn, in part because consideration of NLP as a domain is historically rare in venues that publish most training dynamics research, or analytic work in learning theory. A current search[3] of ICML 2020 publications returned 169 papers with citations to "Association for Computational Linguistics" or "ACL", even including citations to many potential sister conferences: NAACL, AACL, or EACL. A search for citations to a single vision conference, "Computer Vision and Pattern Recognition" or "CVPR", turned up 541 papers. In COLT publications since 2017, the same searches turned up 13 and 23 papers, respectively. In ICML 2020, Wikitext-* or PTB references found only 16 results, while the most popular small corpus for image classification, MNIST, found 264 ICML publications[4].

[3] Searches were performed with Google Scholar.

[4] *CL venues have also become distanced from work in computational *linguistics* (Reiter, 2007), leaving NLP as a field deprived of new scientific work in its data domain as well as new scientific work in its methodologies.

Morcos et al. (2018a) and Raghu et al. (2017) both investigate LSTM behavior, but only tangentially. Raghu et al. (2017) include a plot of PTB (Mikolov et al., 2013b) experiments in an appendix. Morcos et al. (2018a) briefly validate one of their CNN findings with PTB as well as Wikitext-2 (Merity et al., 2017); the NLP equivalent of using the miniscule MNIST (LeCun and Cortes, 2005) dataset in vision. All citations to training dynamics work before 2020 in this chapter focus primarily on CV or image processing tasks.

Linguistics provides us with the salient concept of **markedness** (Andersen, 1989). In language, some forms of a word are the default form, while others are explicitly marked by some additional inflection. An example would be contrast between the word "marked", which is an *unmarked* form compared to "unmarked", which is *marked* by the prefix "un-". In machine learning, we might call CV an *unmarked* domain by convention, in contrast to the *marked* NLP. This convention means that certain tasks and architectures are considered the default environments to understand. Such a convention privileges understanding continuous data over discrete; ConvNets over LSTMs; ResNets over Transformers; geometric tasks over structured prediction.

Understanding one machine learning domain will always extend analysis of others. Latent tree structure is inherent to both domains, but in CV, it is obscured by the image data from which we must compose eyes and mouth into a face—and subsequently, body and face into a cow (Vedaldi et al., 2014). Image classification is a language task, because it is our language that provides the intuitions which we use to construct ontologies that turn into image classes; English does not provide us with common distinctions for different packs of wolves, but it names every dog breed, and so the image labels are chosen according to available terminology. The "default" domain of CV has any number of idiosyncrasies on which to overfit in our understanding of statistical modeling. CV provides us with many interesting geometric phenomena, but the underlying structure of language *without* the added noisy channel of an image can provide a clear and simple domain worth analyzing, as well. A true understanding of statistical models must be a multi-domain understanding, not a mono-domain view focused on one task and its peculiarities.

In order to understand the LSTM architecture better, in this dissertation, we study its behavior on language. We take advantage of the availability of tags for individual words by using them as alternative tasks, finding that LSTMs quickly learn generic input representations and later specialize them to the task at hand (Chapter 4). By

identifying a metric that indicates strong edges within a latent tree structure, we can study how the LSTM prefers language-like tree structures in its representations (Chapter 5). Finally, we use POS tags as a proxy for how predictable a word's behavior is, allowing us to confirm that more predictable words focus on just a few neurons, rather than distributing their gradient (Chapter 6). All of this work takes advantage of well-understood properties of language in order to better understand how an LSTM responds to structured data.

Chapter 4

Beyond Diagnostic Classifiers: Probing Language Model Similarity

> Just as they suspected, the other side of the house looked the same as the front, the back, and the side, and the door was again answered by a man who looked precisely like the other three.
>
> Norton Juster, *The Phantom Tollbooth*

In this chapter, we present the first work focused on training dynamics of modern neural language models. In order to understand how a language model learns underlying properties, such as POS and topic, over the course of training, we first experiment with diagnostic classifiers, a standard probing method. Finding these methods to be insufficiently sensitive to subtle changes in representations over the course of training, we instead introduce a method based on model similarity (Section 2.4.2), comparing the representations produced by language models and models targeting the underlying properties[1].

Using Singular Vector Canonical Correlation Analysis (SVCCA), we compare the similarity of language models (word predictors) with tag predictors at various stages of training. This analysis presents several additional advantages over contemporary probing methods. Training is far more efficient, because it consists only of matrix factorization, rather than training separate neural networks. Furthermore, we do not require parallel annotated data for evaluation, as we are interested only in the internal representations, and not in the performance, of these tag predictors.

[1] We like to call this method the **Similarity Probe for Intrinsic Linguistic Labels** (SPILL).

This strategy yields a variety of informative results about the training process of the language model. We find that coarse POS is learned first, and topic information is learned last, with fine-grained POS and semantic information learned in between. Early in training, models targeting different *tasks* (i.e., language modeling or tag prediction) with the same inputs tend to produce similar representations, and then specialize to their tasks.

We also see that different layers exhibit different behavior: recurrent layer representations become more task-agnostic in late training, but embedding layers become more specialized to their task later in training. However, embedding layers nonetheless remain very generic throughout training when compared to the task-specific recurrent layers. The task-generality of embeddings may explain the effectiveness of pretrained embeddings to initialize representations for other tasks, as a task-agnostic embedding may be used transferred more easily to another environment while upper-level embeddings require more fine-tuning (Howard and Ruder, 2018).

By investigating the language model over the full course of its training, we have a better view of how different layers specialize and the degree to which particular language properties and general input structure shapes representations.

Publication Status This work was published in NAACL 2019.

Understanding Learning Dynamics Of Language Models with SVCCA

Naomi Saphra and **Adam Lopez**
n.saphra@ed.ac.uk alopez@ed.ac.uk
Institute for Language, Cognition, and Computation
University of Edinburgh

Abstract

Research has shown that neural models implicitly encode linguistic features, but there has been no research showing *how* these encodings arise as the models are trained. We present the first study on the learning dynamics of neural language models, using a simple and flexible analysis method called Singular Vector Canonical Correlation Analysis (SVCCA), which enables us to compare learned representations across time and across models, without the need to evaluate directly on annotated data. We probe the evolution of syntactic, semantic, and topic representations and find that part-of-speech is learned earlier than topic; that recurrent layers become more similar to those of a tagger during training; and embedding layers less similar. Our results and methods could inform better learning algorithms for NLP models, possibly to incorporate linguistic information more effectively.

1 Introduction

Large neural networks have a notorious capacity to memorize training data (Zhang et al., 2016), but their high accuracy on many NLP tasks shows that they nonetheless generalize. One apparent explanation for their performance is that they learn linguistic generalizations even without explicit supervision for those generalizations—for example, that subject and verb number agree in English (Linzen et al., 2016); that derivational suffixes attach to only specific parts of speech (Kementchedjhieva and Lopez, 2018); and that short segments of speech form natural clusters corresponding to phonemes (Alishahi et al., 2017). These studies tell us that neural models learn to implicitly represent linguistic categories and their interactions. But *how* do they learn these representations?

One clue comes from the inspection of multilayer models, which seem to encode lexical categories in lower layers, and more contextual categories in higher layers. For example, Blevins et al. (2018) found that a word's part of speech (POS) is encoded by lower layers, and the POS of its syntactic parent is encoded by higher layers; while Belinkov et al. (2018) found that POS is encoded by lower layers and semantic category is encoded by higher layers. More generally, the most useful layer for an arbitrary NLP task seems to depend on how "high-level" the task is (Peters et al., 2018). Since we know that lower layers in a multi-layer model converge to their final representations more quickly than higher layers (Raghu et al., 2017), it is likely that models learn local lexical categories like POS earlier than they learn higher-level linguistic categories like semantic class.

How and when do neural representations come to encode specific linguistic categories? Answers could explain why neural models work and help us improve learning algorithms. We investigate how representations of linguistic structure are learned over time in neural language models (LMs), which are central to NLP: on their own, they are used to produce contextual representations of words for many tasks (e.g. Peters et al., 2018); while *conditional* LMs power machine translation, speech recognition, and dialogue systems. We use a simple and flexible method, Singular Vector Canonical Correlation Analysis (SVCCA; Raghu et al., 2017), which allows us to compare representations from our LM at each epoch of training with representations of other models trained to predict specific linguistic categories. We discover that lower layers initially discover features shared by all predictive models, but lose these features as the LM explores more specific clusters. We demonstrate that different aspects of linguistic structure are learned at different rates within a single recurrent layer, acquiring POS tags early but continuing to learn global topic information later in training.

2 Methods

Our experiments require a LM, tagging models, and a method to inspect the models: SVCCA.

2.1 Language model

Formally, we will model the probability distribution over a sequence of tokens $x_1 \dots x_{|x|}$ with a conventional two-layer LSTM LM. The pipeline from input x_t at time step t to a distribution over x_{t+1} is described in Formulae (1)–(4). At time step t, input word x_t is embedded as (1) h_t^e, which is input to a two-layer LSTM, producing outputs (2) h_t^1 and (3) h_t^2 at these layers, along with cell states c_t^1 and c_t^2. A softmax layer converts h_t^2 to a distribution from which (4) x_{t+1} is sampled.

$$h_t^e = \text{embedding}(x_t) \quad (1)$$
$$h_t^1, c_t^1 = \text{LSTM}_1(h_t^e, h_{t-1}^1, c_{t-1}^1) \quad (2)$$
$$h_t^2, c_t^2 = \text{LSTM}_2(h_t^1, h_{t-1}^2, c_{t-1}^2) \quad (3)$$
$$x_{t+1} \sim \text{softmax}(h_t^2) \quad (4)$$

Each function can be thought of as a *representation* or *embedding* of its discrete input; hence h_t^e is a representation of x_t, and—due to the recursion in (2)—h_t^1 is a representation of $x_1 \dots x_t$.

2.2 Tagging models

To inspect our language model for learned linguistic categories, we will use a collection of tagging models, designed to mimic the behavior of our language model but predicting the next *tag* rather than the next word. That is, given $x_1 \dots x_{|x|}$, we model a corresponding sequence of tags $y_1 \dots y_{|x|}$ using a one-layer LSTM:

$$h_t^{e\prime} = \text{embedding}(x_t) \quad (5)$$
$$h_t^{1\prime}, c_t^{1\prime} = \text{LSTM}(h_t^{e\prime}, h_{t-1}^{1}{}', c_{t-1}^{1}{}') \quad (6)$$
$$y_{t+1} \sim \text{softmax}(h_t^{1\prime}) \quad (7)$$

We will also discuss *input taggers*, which share this architecture but instead sample y_t, the tag of the most recently observed word.

2.3 SVCCA

SVCCA is a general method to compare the correlation of two vector representations. Let d_A and d_B be their dimensions. For N data points we have two distinct views, given by matrices $A \in \mathcal{R}^{N \times d_A}$ and $B \in \mathcal{R}^{N \times d_B}$. We project these views onto a shared subspace in two steps:

1. Use Singular Value Decomposition (SVD) to reduce matrices A and B to lower dimensional matrices A' and B', respectively. This is necessary because many dimensions in the representations are noisy, and in fact cancel each other out (Frankle and Carbin, 2018).

2. Use Canonical Correlation Analysis (CCA) to project A' and B' onto a shared subspace, maximizing the correlation of the projections. Formally, CCA learns matrices W and V to maximize $\rho = \frac{<W^\top A, V^\top B>}{\|wW^\top A\| \|V^\top B\|}$.

Intuitively, the correlation ρ will be high if both representations encode the same information, and low if they encode unrelated information. Figure 1 illustrates how we use SVCCA to compare representation h_t^2 of our language model with the recurrent representation of a tagger, $h_t^{1\prime}$. In practice, we run over all time steps in a test corpus, rather than a single time step as illustrated.

3 Experimental Setup

We trained our LM on a corpus of tokenized, lowercased English Wikipedia (70/10/20 train/dev/test split). To reduce the number of unique words in the corpus, we excluded any sentence with a word type appearing fewer than 100 times. Words appearing fewer than 100 times in the resulting training set are replaced with an unknown token. The resulting training set has over 227 million tokens of 20K types.

We train for 50 epochs to maximize cross-entropy, using a batch size of 40, dropout ratio of 0.2, and sequence length of 35. The optimizer is standard SGD with clipped gradients at 0.25, with the learning rate quartered when validation loss increases. The result of training is shown in Figure 2, which illustrates the dips in loss when learning rate changes.

3.1 Taggers

To understand the representations learned by our LM, we compare them with the internal representations of tagging models, using SVCCA. Where possible, we use coarse-grained and fine-grained tagsets to account for effects from the size of the tagset. Table 1 illustrates our tagsets.

POS tagging For syntactic categories, we use POS tags, as in Belinkov et al. (2017). As a coarse-grained tagset, we use silver Universal Dependency Parse (UDP) POS tags automatically

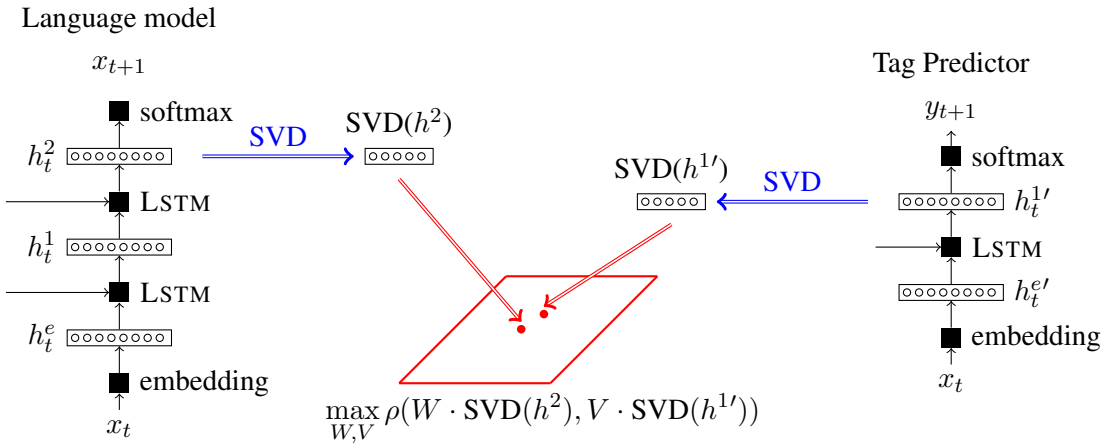


Figure 1: SVCCA used to compare the layer h^2 of a language model and layer $h^{1\prime}$ of a tagger.

Tag	These	cats	live	in	that	house	.
UDP POS	DET	NOUN	VERB	ADP	DET	NOUN	SYM
PTB POS	DT	NNS	VBP	IN	DT	NN	.
SEM (coarse)	DEM	ENT	EVE	ATT	DEM	ENT	LOG
SEM (fine)	PRX	CON	ENS	REL	DST	CON	NIL
topic	1	1	1	1	1	1	1

Table 1: An example sentence annotated with all tags, assuming its source is an article with document ID of 1.

Figure 2: Test performance of the LM. Vertical dotted lines indicate when the optimizer rescale the step size.

added to our Wikipedia corpus with spacy.[1] We also use a corpus of fine-grained human annotated Penn Treebank POS tags from the Groningen Meaning Bank (GMB; Bos et al., 2017).

Semantic tagging We follow Belinkov et al. (2018) in representing word-level semantic information with silver SEM tags (Bjerva et al., 2016). SEM tags disambiguate POS tags in ways that are relevant to multilingual settings. For example, the comma is not assigned a single tag as punctuation, but has distinct tags according to its function: conjunction, disjunction, or apposition. The 66 fine-grained SEM tag classes fall under 13 coarse-grained tags, and an 'unknown' tag.

[1] https://spacy.io/

Global topic For topic, we classify the each word of sequence by its source Wikipedia article; for example, every word in the wikipedia article on Trains is labeled "Trains". This task assesses whether the network encodes the global topic of the sentence.

UDP silver POS and topic information use the same corpus, taken from the 100 longest articles in Wikipedia in a 70/10/20 train/dev/test split. The corpus is taken from the LM training data, which may increase the similarity between the tag model and LM. Because both tag predictors are trained and tested on the same domain as the LM, they can be easily compared in terms of their similarity to the LM representation. Though the SEM corpus and the PTB corpus are different domains from the Wikipedia training data, we compare activations on the same 191K-token 100-article test corpus.

Table 2 describes the training and validation corpus statistics for each tagging task. Note that topic and UDP POS both apply to the same en-wikipedia corpus, but PTB POS and SEM use two different unaligned sets from the GMB corpus.

4 Experiments, Results, and Analysis

A benefit of SVCCA is its flexibility: it can be used to compute correlations of a hidden represen-

tag	corpus	number of classes	token count train	token count dev	label $t+1$ acc	label $t+1$ ppl	label t acc	label t ppl	randomized acc	randomized ppl
UDP POS	wiki	17	665K	97K	50	4.3	93	1.2	21	8.9
PTB POS	GMB	36	943K	136K	51	4.7	95	1.18	14	18.0
SEM (coarse)	GMB	14	937K	132K	55	3.5	91	1.3	22	9.0
SEM (fine)	GMB	67	937K	132K	50	5.6	88	1.45	17	21.5
topic	wiki	100	665K	97K	36	19.1	37	16.3	5	81.5

Table 2: Tag predictor and tagger statistics. Accuracy and perplexity on $t+1$ are from the target tag predictor, on t are from the input tagger. Metrics obtained when training on randomly shuffled labels are provided as a low baseline. Accuracy is on the test set from the training domain (GMB or Wikipedia).

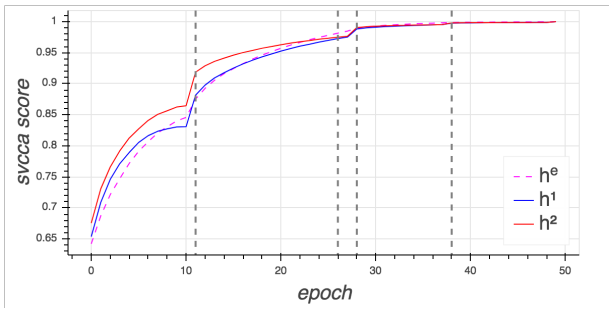


Figure 3: SVCCA score between representations at each epoch and from the final trained LM.

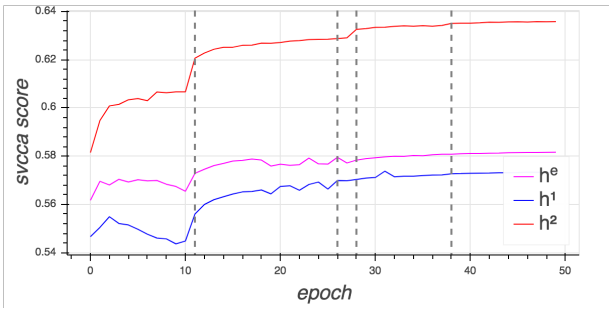


Figure 4: SVCCA score between the LM at each epoch and a LM with different initialization.

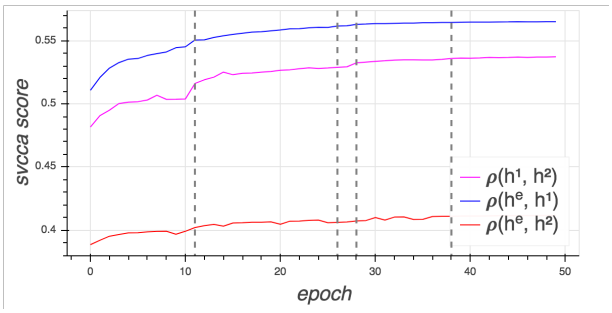


Figure 5: SVCCA score between different layers of the LM at each epoch. For example, $h_t^2 - h_t^1$ compares the activations h_t^2 with the activations h_t^1 after epoch t.

tation with any other vector. Raghu et al. (2017) used it to understand learning dynamics by comparing a learned representation to snapshots of the same representation at different epochs during training. We use a similar experiment to establish the basic learning dynamics of our model. In our shallow 2-level model, activations at h^1 converge slightly after h^2 (Figure 3). This differs from the results of Raghu et al. (2017), who found that a 5-layer stacked LSTM LM exhibits faster convergence at lower layers, but this difference may be attributed to our much larger training data, which our model fits more accurately in fewer epochs.

Empirical upper bounds. Our main experiments will test the rate at which different linguistic categories are learned by different layers, but to interpret the results, we need to understand the behaviour of SVCCA for these models. In theory, SVCCA scores can vary from 0 for no correlation to 1 for perfect correlation. But in practice, these extreme cases will not occur. To establish an *empirical* upper bound on correlation, we compared the similarity at each epoch of training to the frozen final state of a LM with identical architecture but different initialization, trained on the same data (Figure 4).[2] The correlations increase over time as expected, but to a maximum near 0.64; we don't expect correlations between our LM and other models to exceed this value. We explore corresponding lower bounds in our main experiments below.

Correlations between different layers. Next we examine the correlation between different layers of the same model over time (Figure 5). We observe that, while over time correlation increases, in general closer layers are more similar, and they are less correlated than they are with the same layer of a differently initialized model.

SVCCA vs. Diagnostic classifiers A popular method to analyze learned representations is to use a *diagnostic classifier* (Belinkov et al., 2017), a

[2] This experiment is similar to the comparisons of randomly initialized models by Morcos et al. (2018).

separate model that is trained to predict a linguistic category of interest, y_t, from an arbitrary hidden layer h_t. Diagnostic classifiers are widely used (Belinkov et al., 2018; Giulianelli et al., 2018). But Zhang and Bowman (2018) found that if a diagnostic classifier is trained on enough examples, then random embeddings as input representations often outperform any pretrained intermediate representation. This suggests that diagnostic classifiers may work simply by memorizing the association between an embedding and the most frequent output category associated with that embedding; since for many words their category is (empirically) unambiguous, this may give an inflated view of just how much a model "understands" about that category.

Our use of SVCCA below will differ from the use of diagnostic classifiers in a couple of important ways.

1. Diagnostic classifiers use the intermediate representations of the LM as inputs to a tagger. A representation is claimed to encode, for example, POS if the classifier accurately predicts it—in other words, whether it can *decode* it from the representation. We will instead evaluate the *similarity* between the representations in an LM and in an independently-trained tagger. The intuition behind this is that, if the representation of our LM encodes a particular category, then it must be similar to the representation of model that is specifically trained to predict that category. A benefit of this is that the similarity can be evaluated on *any* dataset, not only one that has been labeled with the linguistic categories of interest.

2. Typically, diagnostic classifiers are used to decode tag information about the context or most recent *input* from the hidden state at the current step. Because the hidden representation at time t is meant to encode predictive information about the target word at time $t+1$, we treat it as encoding a prediction about the tag of the *target* word.

To understand the empirical strengths and weaknesses of these approaches, we compare the use of SVCCA and diagnostic classifiers in understanding learning dynamics. In other words, we ask: is our first conceptual shift (to SVCCA) necessary? To test this, we use the same model as Belinkov et al. (2017), which classifies an arbitrary representation using a ReLU followed by a softmax layer. To be consistent with Belinkov et al. (2017), we use y_t as their target label. We repeat their method in this manner (Figure 6) as well as applying our second modification, in which we instead target the label y_{t+1} (Figure 7).

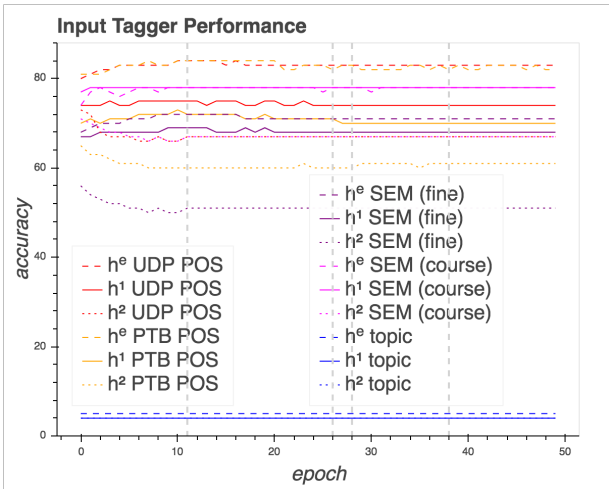


Figure 6: Learning dynamics interpreted with diagnostic classifiers labeling input word tag y_t.

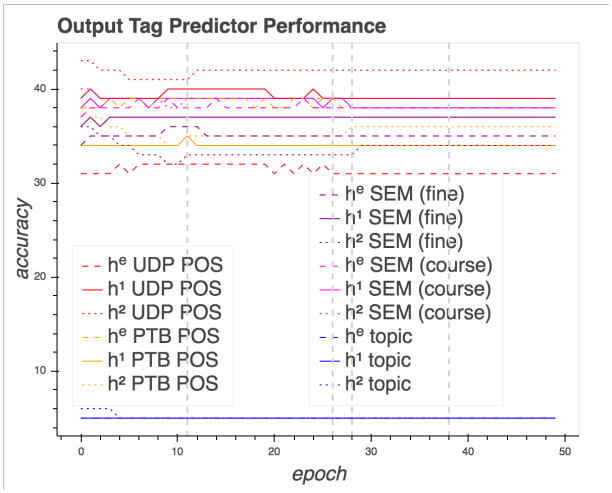


Figure 7: Learning dynamics interpreted with diagnostic classifiers labeling target word tag y_{t+1}.

We found the correlations to be relatively stable over the course of training. This is at odds with the results in Figures 2 and 3, which suggest that representations change substantially during training in ways that materially affect the accuracy of the LM. This suggests that diagnostic classifiers are indeed learning associations between embeddings and output classes, and we conclude that they are ineffective for understanding learning dynamics. Our remaining experiments use only SVCCA.

4.1 SVCCA on Output Tag Prediction

We applied SVCCA to each layer of our LM with the corresponding layer of each tag predictor in

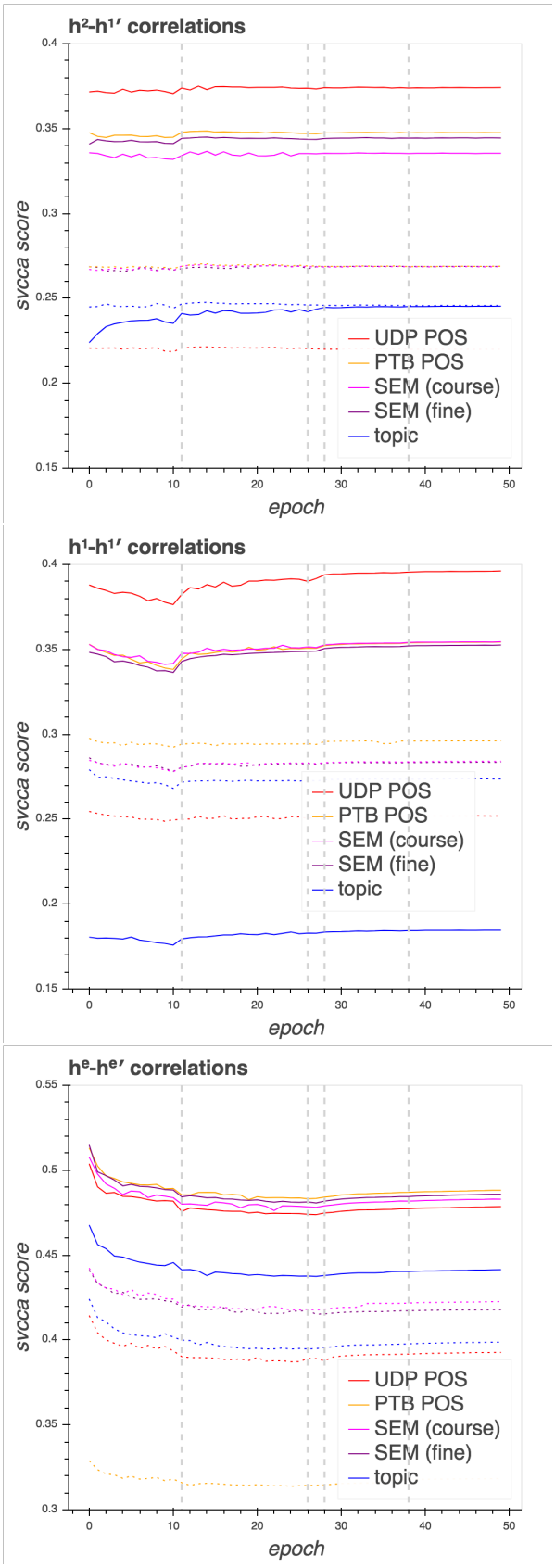


Figure 8: SVCCA correlation scores between the LM predicting x_{t+1} and the tag model predicting y_{t+1}. At the end of each epoch, we compare the current LM with the final tag model. Dotted lines use shuffled tags. Gray vertical lines mark when the step size is rescaled.

order to find the correlation between the LM representation and the tag model representation at each level (Figure 8). To establish empirical lower bounds on correlation, we also trained our taggers on the same data with randomly shuffled labels, as in Zhang et al. (2016). These latter experiments, denoted by the dotted lines of Figure 8, show how much of the similarity between models is caused by their ability to memorize arbitrary associations.

The strongest similarity at recurrent layers belongs to the most local property, the UDP POS tag. Both coarse- and fine-grained semantic tags, which rely on longer range dependencies, fall below UDP POS consistently. Topic, which is global to an entire document, is the least captured and the slowest to stabilize. Indeed, correlation with true topic falls consistently *below* the score for a model trained on randomized topic tags, implying that early in training the model's representation does not capture enough context to identify topic, which depends on sets of words rather than individual words. Over time correlation improves, possibly because the model encodes long-distance context. Khandelwal et al. (2018) found that LSTMs remember content words like nouns for more time steps than they remember function words like prepositions and articles. We hypothesize that the LM's slower stabilization on topic is related to this, since it must depend on content words, and its ability to remember them increases throughout training.

The encoder layer exhibits very different patterns. Because the representation produced by the encoder layer is local to the word, the nuances that determine how a word is tagged in context cannot be learned. To the contrary, similarity between the encoders declines over time as they improve in their specialization. This decline points to some easily learned patterns which are helpful for all tasks, but which are gradually replaced by representations more useful for language modeling. This process may even be considered a naturally occurring analog to the common practice of initializing the encoder layer as word embeddings pretrained an unrelated task such as skip-gram or CBOW (Mikolov et al., 2013). It seems that the 'easy' word properties which immediately improve performance are similar regardless of the particular language task.

The encoder layers are all highly similar to each other, which suggests that the unigram representa-

tions produced by the encoder are less dependent on the particular end task of the neural network. This also fits well with the literature on word embeddings, where we find that pretraining the encoder on unrelated tasks significantly improves performance on many natural language problems.

At h^1, the correlation shows a clear initial decline in similarity for all tasks. This seems to point to an initial representation that relies on simple shared properties, which in the first stage of training is gradually dissolved before the layer begins to converge on a structure shared with each tag predictor. It may also be linked to the information bottleneck learning phases explored by Shwartz-Ziv and Tishby (2017). They suggest that neural networks learn by first maximizing the mutual information between the input and internal representation, then minimizing the mutual information between the internal representation and output. The network thus initially learns to effectively represent the input, then compresses this representation, keeping only the elements relevant to the output. If the LM begins by maximizing mutual information with input, because the input is identical for the LM and tag models it may lead to these similar initial representations, followed by a decline in similarity as the compression targets properties specific to each task.

4.2 SVCCA on Input Tagging

Our second conceptual shift is to focus on output tag prediction—asking what a representation encodes about the *next* output word, rather than what it has encoded about words it has already observed in the input. What effect does this have? Since we already studied output tags in the previous set of experiments, here we consider input tags, in the style of diagnostic classifier analysis (Figure 9). The learning dynamics are simliar to those for tag prediction, but the UDP POS tagger decreases dramatically in all correlations while the GMB-trained taggers (PTB POS, SEM (fine), and SEM (coarse)) often increase slightly. While the shapes of the lines are similar, UDP POS no longer consistently dominates the other tasks in recurrent layer correlation. Instead, we find the more granular PTB POS tags lead to the most similar representations.

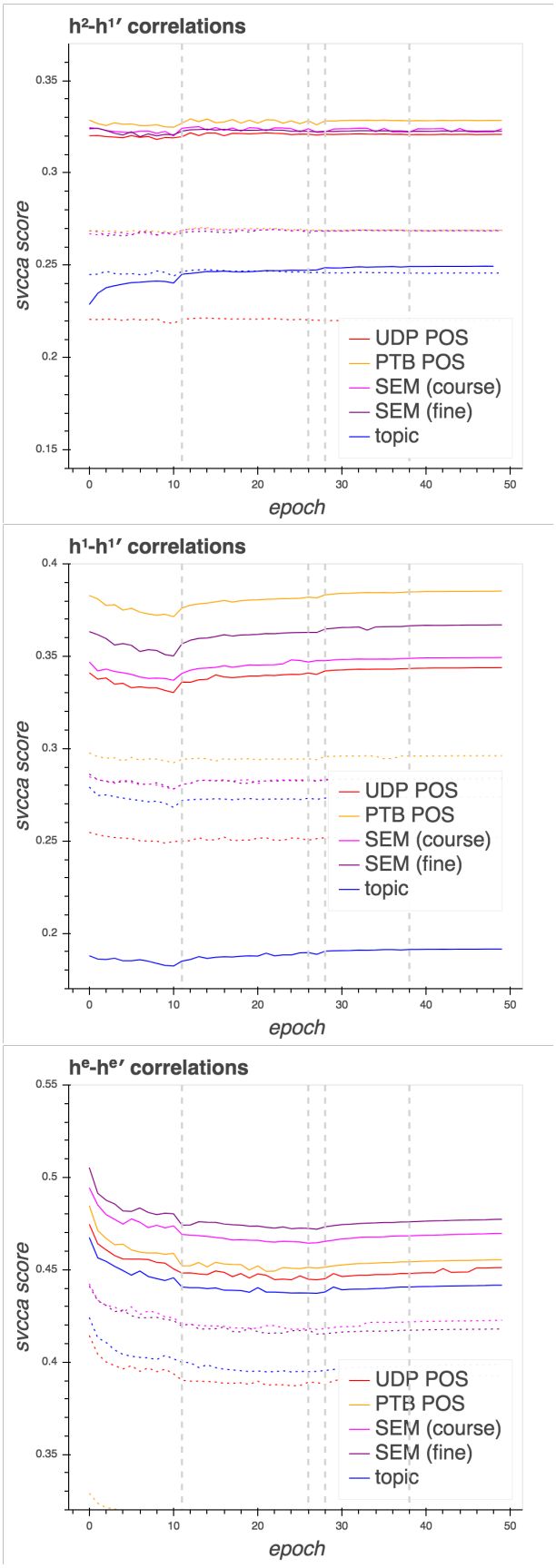


Figure 9: SVCCA correlation scores between LM activations when predicting x_{t+1} and tagger activations when labeling y_t. Dotted lines use shuffled tags. Gray vertical lines mark when the step size is rescaled.

5 Discussion and Conclusions

We find clear patterns in the encoding of linguistic structure with SVCCA, in contrast to the weaker results from a less responsive diagnostic classifier. Because SVCCA proves so much more sensitive than the diagnostic classifiers currently in use, we believe that future work on measuring the encoding of linguistic structure should use the similarity of individual modules from independently trained tag predictors rather than the performance of tag predictors trained on a particular representation.

This system should also be of interest because it is efficient and convenient. To train a diagnostic classifier, we must run a forward pass of the LM for each forward pass of the auxiliary model. With B as the batch size and N, D, and T as the respective sizes of the training, development, and test set, we must perform $O(\frac{N+D+T}{B})$ forward pass timesteps of the LM in order to train diagnostic classifiers for a single tag set. Because our tag models are trained independently for SVCCA, we only run the LM on the tag test set, $O(\frac{T}{B})$ times in total. With our implementation, this difference in complexity was expressed as SVCCA experiments running in hours instead of the days required for a diagnostic classifier experiment.

Our method holds another, more subtle advantage. Our analysis tests a specific assumption about how structure might be encoded within a LM. If the model's predictions rely on implicitly represented linguistic categories, then its internal representation should correlate with the representation in an explicit model of those categories. Moreover, this correlation will be layerwise, since, as we have seen, different layers encode different information. But the use of diagnostic classifiers does not reflect how each layer expects to interact with the model as a whole.

What do we learn about the LM when a feedforward network cannot extract tag information directly from the embedding layer, but can from a recurrent layer? It may be tempting to conclude that tag information relies heavily on context, but if the embedding encodes the tag to be interpreted by a recurrent layer, a feedforward network may not be capable of representing the function to extract that tag because it does not have access to a context vector for aiding interpretation of the hidden layer, or because its activation functions cover a different range. By directly comparing LSTM layers to LSTM layers and embedding layers to embedding layers, we respect the role of each module within the network in our analysis.

The results of our analysis imply that early in training, representing part of speech is the natural way to get initial high performance. However, as training progresses, it increasingly benefits the model to represent categories with longer-range dependencies, such as topic.

6 Future Work

One direction for future work is exploring how generalization interacts with the correlations between LMs and tag predictors. It may be that a faithful encoding of a property like POS tag indicates that the LM is relying more on linguistic structure than on memorizing specific phrases, and therefore is associated with a more general model.

If these measurements of structure encoding are associated with more general models, we might introduce regularizers or other modifications that explicitly encourage correlation with a tagging task.

Combes et al. (2018) identified the phenomenon of *gradient starvation*, meaning that while frequent and unambiguous features are learned quickly in training, they slow down the learning of rarer features. For example, artificially brightening images according to their class leads to a delay in learning to represent the less consistent natural class features. Although it is tempting to claim that semantic structure is learned using syntactic structure as natural scaffolding, it is possible that the simple predictive power of POS is acting as an attractor and starving semantic features that are rarer and more ambiguous. A possible direction for future work would be to explore which of these explanations is true, possibly by decorrelating particular aspects of linguistic structure from language modeling representations.

The techniques in this paper could be applied to better understand the high performance of a system like ELMo (Peters et al., 2018). Different layers in such a system are useful for different tasks, and this effect could be understood in terms of the gradual divergence between the layers and their respective convergence to representations geared toward a single task.

Acknowledgements

We thank Sameer Bansal, Toms Bergmanis, Maria Corkery, Sharon Goldwater, Sorcha Gilroy, Aibek Makazhanov, Yevgen Matusevych, Kate Mc-

Curdy, Janie Sinclair, Ida Szubert, Nikolay Bogoychev, and Clara Vania for helpful discussion and comments on drafts. We thank Matthew Summers for assistance with visualisations.

References

Afra Alishahi, Marie Barking, and Grzegorz Chrupała. 2017. Encoding of phonology in a recurrent neural model of grounded speech. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 368–378.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do Neural Machine Translation Models Learn about Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2018. Evaluating Layers of Representation in Neural Machine Translation on Part-of-Speech and Semantic Tagging Tasks. *CoRR*, abs/1801.07772.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic Tagging with Deep Residual Networks. In *COLING*.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. *arXiv:1805.04218 [cs]*. ArXiv: 1805.04218.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In *Handbook of linguistic annotation*, pages 463–496. Springer.

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.

Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Training Pruned Neural Networks. *arXiv:1803.03635 [cs]*. ArXiv: 1803.03635.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the Hood: Using Diagnostic Classifiers to Investigate and Improve how Language Models Track Agreement Information. *arXiv:1808.08079 [cs]*. ArXiv: 1808.08079.

Yova Kementchedjhieva and Adam Lopez. 2018. *Indicatements* that character language models learn English morpho-syntactic units and regularities. In *Proc. of Workshop on Analyzing and interpreting neural networks for NLP*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. *arXiv:1805.04623 [cs]*. ArXiv: 1805.04623.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *arXiv:1611.01368 [cs]*. ArXiv: 1611.01368.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. *arXiv:1806.05759 [cs, stat]*. ArXiv: 1806.05759.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. *arXiv:1706.05806 [cs, stat]*. ArXiv: 1706.05806.

Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the Black Box of Deep Neural Networks via Information. *arXiv:1703.00810 [cs]*. ArXiv: 1703.00810.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Kelly W. Zhang and Samuel R. Bowman. 2018. Language Modeling Teaches You More Syntax than Translation Does: Lessons Learned Through Auxiliary Task Analysis. *arXiv:1809.10040 [cs]*. ArXiv: 1809.10040.

A Performance Out Of Domain

Because SEM tags and PTB POS tags were both trained on the GMB corpus, we present the SVCCA similarities on an in-domain GMB test corpus as well as the Wikipedia test corpus used elsewhere in the paper. The results are in Figures 10-11. In general correlations are higher using the original tagging domain, but not enough to contradict our earlier analysis. The shapes of the curves remain similar.

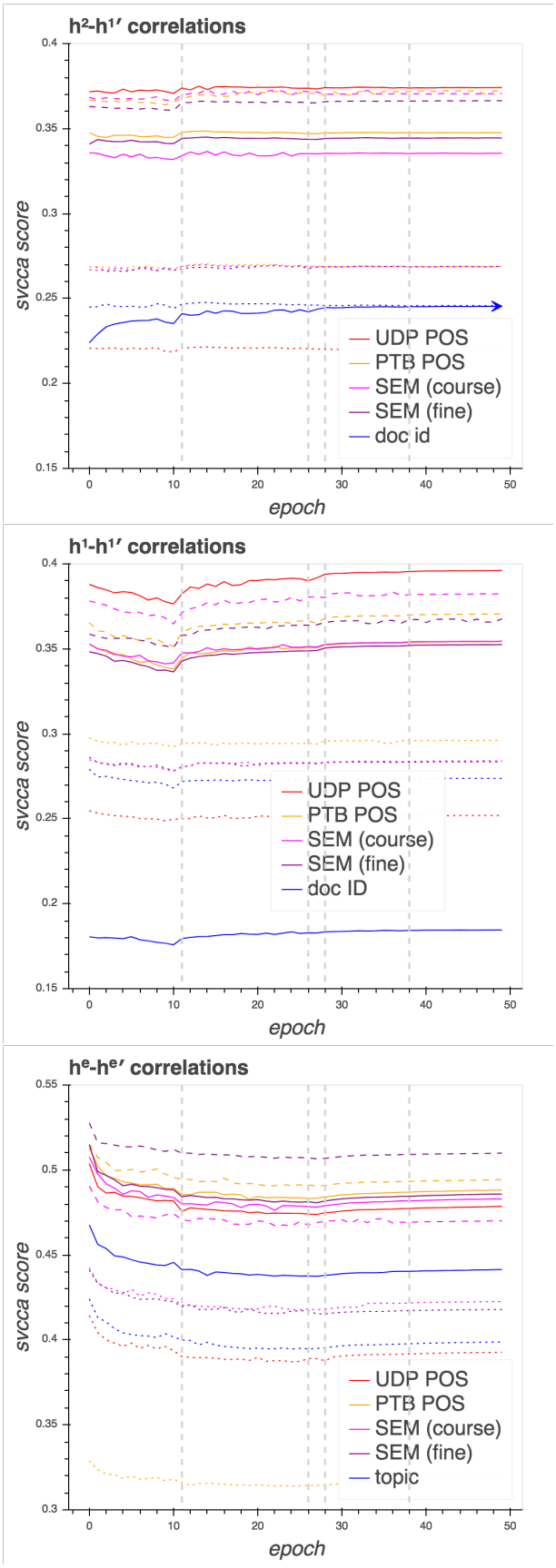


Figure 10: SVCCA correlation scores between LM and y_{t+1} tag predictor. Dotted lines use models trained on randomly shuffled the data. Dashed lines use GMB domain test data.

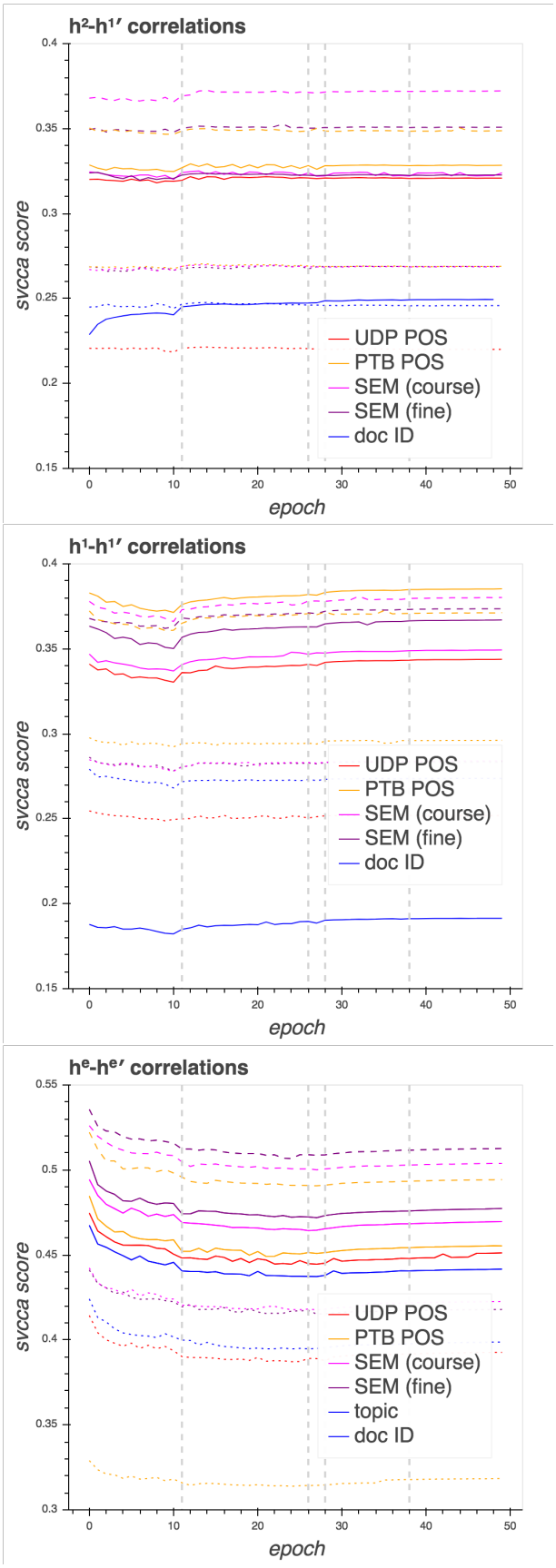


Figure 11: SVCCA correlation scores between LM and y_t tagger. Dotted lines use models trained on randomly shuffled the data. Dashed lines use GMB domain test data.

4.1 Comments on the paper

In addition to the insights we gain from observing the time course of training, the model similarity method we use provides some intriguing insights even without considering the time dimension. When looking at input tagging instead of next-tag prediction, we find that more granular tagging produces representations that are more similar to LM representations (e.g., a model predicting simple Universal Dependencies POS tags produces more LM-like representations than a model predicting more granular PTB POS tags). This result suggests that complex tag information is retained from input words, but the structure ultimately encoded in predictions does not resemble the linguistic ontologies. We also see that input structure itself is retained, even given random target labels, based on the high baseline similarity between the language model and randomized (that is, with *memorized* outputs) taggers. Results like this suggest *general* properties of the LSTM, and possibly further of neural networks, in their reliance on input structure independent of output.

These insights suggest the possibility of using similarity as a general probing method, outside of training dynamics experiments–and indeed, since the publication of this paper, representational similarity methods have been widely applied in NLP. Chrupala and Alishahi (2019) and Chrupala (2019) both explored the correlations between different vector and symbolic representations using Representational Similarity Analysis. Movva and Zhao (2020) and Bau et al. (2019) used neuron-level similarity to compare models. Voita et al. (2019a) looked at the similarity between different layers of a Transformer model using a different variant on CCA, PWCCA. Singh et al. (2019b), meanwhile, compared representations in different languages using unadorned CCA, and Hsu et al. (2019) used SVCCA as in our paper. Chung et al. (2020) and Wu et al. (2020a) both surveyed a variety of similarity techniques, with Wu et al. (2020a) validating one of the assumptions behind our method by confirming that similar architectures produced similar representations. Note that nearly all of these works postdate the work in this chapter, and almost all of them cite it.

While in general, models are more accurate when they are more similar to the same architecture trained with different seeds (Raghu et al., 2017), it is yet to be seen whether high similarity *across tasks* (e.g., LM compared to a tagging task as in this paper) indicates higher LM performance. If so, it would be interesting to see *which tasks*. Can we glean information about LM performance from similarity to representations

for all tasks, or only for those that involve a closer distance of dependencies (POS) or farther (topic)? This is an essential result to confirm that similarity analysis avoids one of the flaws of diagnostic classifiers: a lack of correlation between model accuracy and probe accuracy.

It is worth noting that there is no theoretical reason establishing that modules should have high similarity scores, even if they are the same type of module and situated with a similar environment in the pipeline of a model, but in practice (Chung et al., 2020; Raghu et al., 2017; Morcos et al., 2018a) this does seem to be the case. Therefore, although CCA yields specifically a *linear* similarity, it is distinct from the common use of linear functions as probing models in that there is a clear empirical justification for the assumption that two representations will be linearly similar, if they are accomplishing the same function (such as representing POS as a recurrent module feeding into a softmax layer).

4.1.1 Generalizing to Attentional Models

The methods of this paper could be straightforwardly applied to any Transformer-based LM, including masked LMs rather than autoregressive. In order to do so, one would substitute the desired tag for the word being predicted and apply SVCCA to the activation vectors produced by, e.g., the feedforward layers of each Transformer module. In addition, similarity of the attention distributions used by models predicting a tag vs. a word could be measured through Shannon- or KL-divergence.

Chapter 5

Beyond Probing: The Development of Hierarchical Construction

> "Oh no," said Milo seriously. "In my family we all start on the ground and grow up, and we never know how far until we actually get there."
> "What a silly system." The boy laughed. "Then your head keeps changing its height and you always see things in a different way"
>
> Norton Juster, *The Phantom Tollbooth*

In Chapter 4, we saw that an LSTM quickly learns properties like POS tags which depend mostly on local context. But it's not so quick to learn topic tags, which depend on more distant context in the document; instead, the LSTM seems to initially *remove* topic information. In general, this model architecture learns short-distance associations faster than long-distance ones, but does it *depend* on short-distance associations to build the longer-distance ones?

This paper describes how hierarchical syntactic and linguistic structures evolve in an LM: in these synthetic environments, they actually emerge through a soft chunking process. That is, we show that throughout training, *the LSTM learns stereotyped new trees out of the subtrees it has already learned, using those subtrees as scaffolding.* The resulting theory of hierarchical construction goes beyond the tracking of broad properties like model similarity or probe performance over the course of training. Instead, we propose a specific local process that leads to compositional behavior. This local process entails chunks phrase meanings together, so the meaning of words depends

increasingly on familiar neighbors. We quantify this **interdependence** according to the difference between the isolated representations of phrases (computed with Contextual Decomposition) and their combined meanings. We inspect the interdependence between two different phrases in a synthetic environment to view how familiar constituents influence the training of their neighbors[1].

Publication Status This work was published in Findings of EMNLP 2020.

[1] In the paper this is described as **Decompositional Interdependence**. We prefer to call it **Semantic Interdependence Summary**, or SIS.

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).

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:

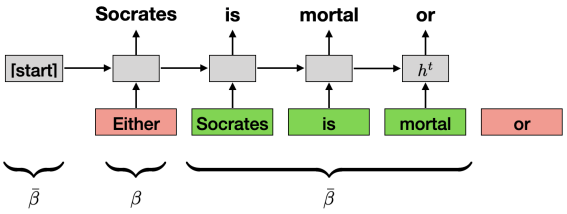


Figure 1: CD uses linear approximations of gate operations to linearize the sequential application of the LSTM module. CD produces the vector h^t_β isolating the contribution of "Either" to the vector h^t predicting "or", as well as producing the irrelevant contribution $h^t_{\bar{\beta};\beta\bowtie\bar{\beta}}$. 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.

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

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]

[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 $\beta \reflectbox{~} \bar{\beta}$, 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

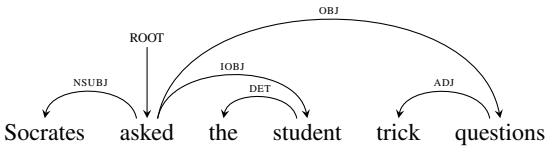


Figure 2: A dependency parsed sentence.

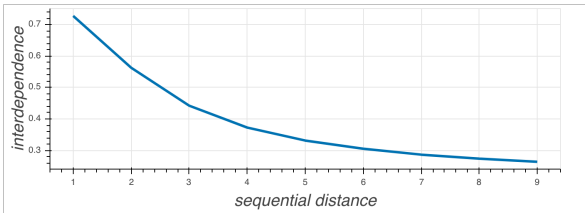


Figure 3: Average DI between word pairs x_l, x_r at different sequential distances $r - l$.

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, $t = \max(\text{idx}(A), \text{idx}(B))$.

[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.

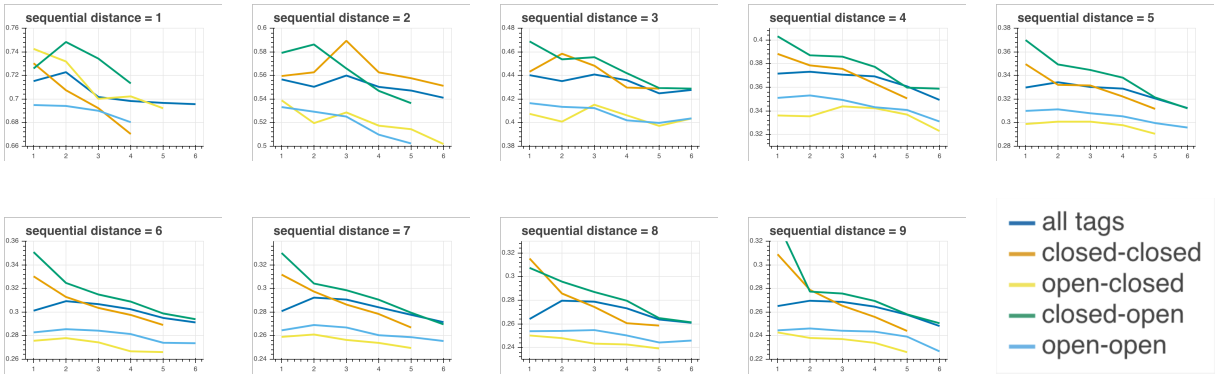


Figure 4: Mean *DI* (y-axis) between word pairs at varying *syntactic distances* (x-axis), stratified by *whether the POS tags are closed or open class* (line color) and by *sequential distance* (plot title). The y-axis ranges differ, but the scale is the same for all plots. Each mean is plotted only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

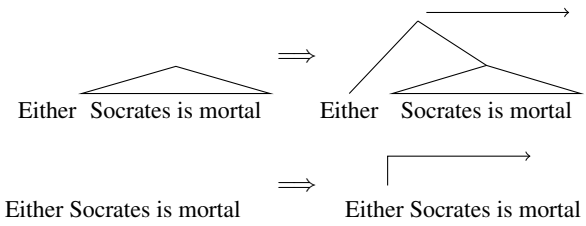


Figure 5: Top: A familiar span (indicated by a triangle illustrating it as a recognizable constituent) is used as a scaffold in its new context, allowing the model to construct a closely interdependent representation for predicting the next word. Bottom: An unfamiliar span cannot be used as a scaffold,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

47 306 71 287 87 660 892 645 768 472 332 482 171 786
833 271 595 421 527 392 328 648 581 709 549 107 279 115
324 595 467 352 23 484 388 543 996 176 8 748 677 833
137 467 852 4 450 830 178 535 743 339 903 48 170 891
657 379 704 791 516 255 782 173 384 88 723 975 498 698
181 562 14 35 139 881 (((424 363 677 173 404 381 443
145))) 685 700 995 391 791 153 710 615 590 996 168 897
943 981 156 438 726 146 754 671 92 657 521 467 822 100

(a) unfamiliar-scaffold training set

47 306 71 424 363 677 173 404 381 443 145 482 171 786
833 271 595 421 527 392 328 648 581 709 549 107 279 115
324 595 467 352 23 484 388 543 424 363 677 173 404 381
443 145 852 4 450 830 178 535 743 339 903 48 170 891
657 379 704 791 516 255 782 173 384 88 723 975 498 698
181 562 14 35 139 881 (((424 363 677 173 404 381 443
145))) 685 700 995 391 791 153 710 615 590 996 168 897
943 981 156 438 726 146 754 671 92 657 521 467 822 100

(b) familiar-scaffold training set

370 292 935 528 567 662 464 964 758 331 480 808 152 23
454 (((432 729 356 669 260 434 364 757))) 166 900 223
122 339 25 116 989 662 24 983 323 110 82 370 961 508
280 638 934 752 583 23 425 919 216 654 538 805 808 646
32 276 585 303 849 21 372 376 568 906 896 (((215 102
614 389 485 454 723 16))) 352 546 408 983 578 847 15
953 992 844 410 63 104 283 115 218 879 0 223 596 138
822 361 125 28 74 (((79 906 922 607 667 784 548 988)))

(c) out-domain test set

370 292 935 528 567 662 464 964 758 331 480 808 152 23
454 (((424 363 677 173 404 381 443 145))) 166 900 223
122 339 25 116 989 662 24 983 323 110 82 370 961 508
280 638 934 752 583 23 425 919 216 654 538 805 808 646
32 276 585 303 849 21 372 376 568 906 896 (((424 363
677 173 404 381 443 145))) 352 546 408 983 578 847 15
953 992 844 410 63 104 283 115 218 879 0 223 596 138
822 361 125 28 74 (((424 363 677 173 404 381 443 145)))

(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.

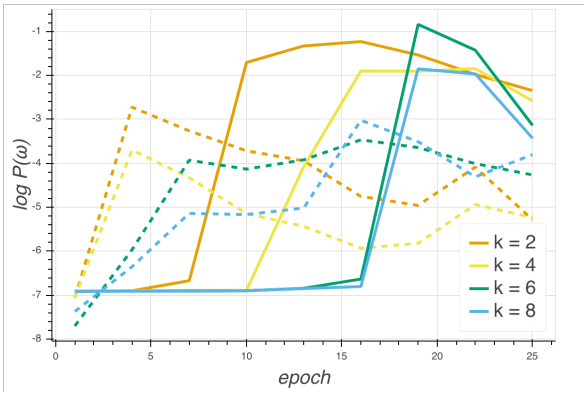
4.3 The Effect of Scaffold Familiarity

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.

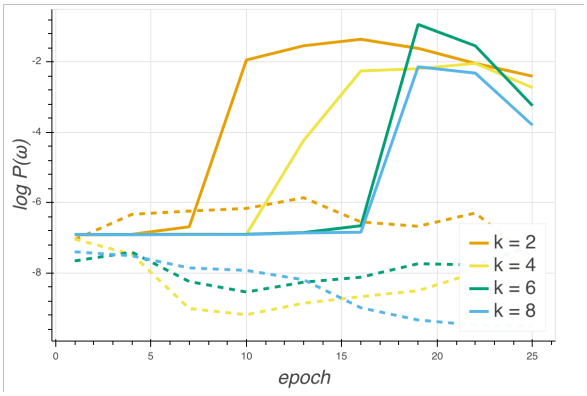
In the **familiar-scaffold setting**, we randomly distribute 1000 occurrences of each scaffold throughout the corpus outside of the rule patterns. Therefore each scaffold is seen often enough to be memorized (see Appendix B). In the original **unfamiliar-scaffold setting**, q appears only as a scaffold, 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.

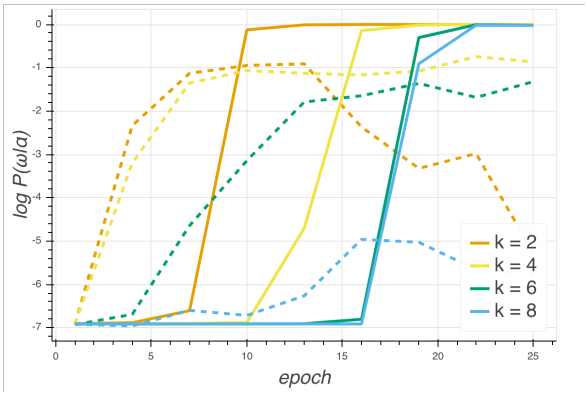


Figure 8: Mean target probability of ω at its correct timestep based on CD with α in focus, on out-domain test set. Solid lines are trained in the unfamiliar-scaffold set, dashed lines on familiar-scaffold.

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-

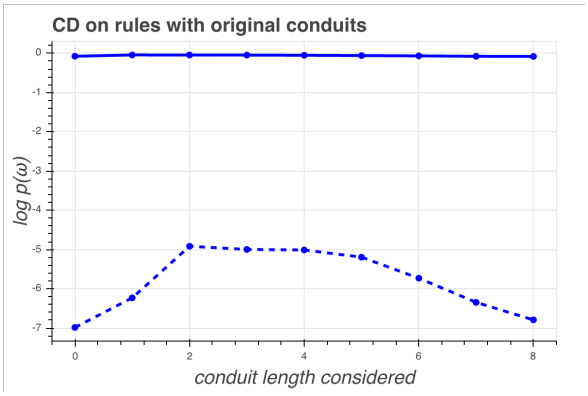


Figure 9: The predicted $P(x_t = \omega | x_{t-k} \dots x_{t-k+i})$ according to CD, varying i as the x-axis and with $x_{t-k} = \alpha$ and $k = 8$. Solid lines are trained in the unfamiliar-scaffold set, dashed lines on familiar-scaffold.

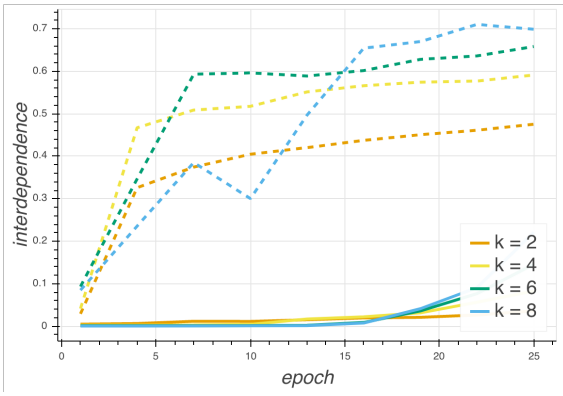


Figure 10: Mean DI(α, scaffold) on the in-domain test set. Solid lines are trained in the unfamiliar-scaffold set, dashed lines on familiar-scaffold.

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.

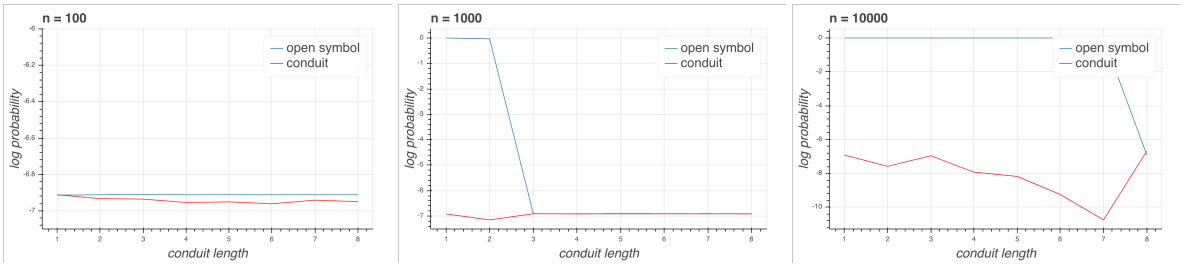


Figure 11: The predicted probability $P(x_t = \omega)$, according to the contributions of open symbol $x_{t-k} = \alpha$ and of the scaffold sequence $x_{t-k+1} \ldots x_{t-1}$, for various rule occurrence counts n. Shown at 40 epochs.

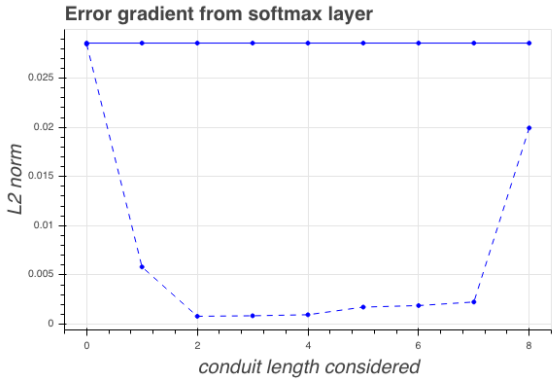


Figure 12: Average gradient magnitude ΔE_{t+-k+d}, varying d up to the length of the scaffold. Solid lines are the unpredictable scaffold setting, dashed lines are the predictable scaffold setting.

5.1 Comments on the Paper

In this paper, we have treated **bottom-up composition**, in contrast with top-down composition, similarly to traditional work in parsing (Jurafsky and Martin, 2009, pp 428-432). That is, we show how the LSTM chunks syntactic constituents together, iteratively building into larger trees, rather than learning larger tree patterns and splitting them, or executing some idiosyncratic haphazard system of learning common connections which somehow end up similar to syntax. We measure this compositional behavior by measuring how interdependent words are, which is correlated with syntactic proximity.

Linguistics has a long history of argument over the nature of compositionality, considering the conflicting nature of bottom-up and top-down constructions of meaning (Herbelot, 2020). First, Frege's Principle points to bottom-up construction of meaning from repeated joining of constituents, declaring that *"the meaning of the whole sentence is a function of the meanings of its parts"* (Cresswell, 1973). This leads to the localism and resulting tree structure we analyze here, but there are other aspects to what we call "compositionality" (Hupkes et al., 2020). The top-down view is a recognition that no word has meaning independent of its context—which is also fundamental to the measure of interdependence, as it depends on the impossibility of separating out any word's meaning entirely from the nearby words it interacts with.

We chose to consider how interdependence reveals bottom-up construction, but we could just as easily have analyzed interdependence as a lens into the top-down construction of meaning through context. Future work, for example, may consider stereotyped dialog (like exchanges of greetings), coreference, or multiword expressions (Schneider and Smith, 2015) with intervening phrases. By explicitly comparing how closely interdependence matches with syntax trees rather than coreferences, we ask which of these underlying structures the model is leveraging to learn language.

5.1.0.0.1 Relation to Chapter 4: Saphra and Lopez (2019) illustrated that some linguistic properties based on short range dependencies, such as POS tags, are learned earlier than properties requiring more context, especially topic information. In this chapter, we suggest a possible explanation: that an LSTM uses shorter constituents as scaffolding for longer dependencies. However, we don't present any causal evidence. In order to make a causal claim, we would need to interfere with this scaffolding phenomenon and observe its effect on the order in which properties are learned, according

to the SVCCA probe.

5.1.1 Expansions of Contextual Decomposition

In future work, we may question whether Contextual Decomposition (Murdoch et al., 2018) is the best approach to compute phrase importance. Since Murdoch et al. (2018), a number of expansions to CD have been published (Jin et al., 2020; Singh et al., 2019a; Jumelet et al., 2019). Jin et al. (2020) add a sampling step for nearby words in **Sampling and Contextual Decomposition** (SCD). SCD adheres better to two preferences in its output: **context independence** and **non-additivity**. Context independence is a preference for credit attribution that is independent of a phrase's context in a sentence. For example, in the sentence "Socrates asked a trick question," we should see "trick question" retain the same meaning in any other context. This property would mitigate the change in meaning as we shift to a novel domain, which instead we consider as part of our analysis in the paper's Figure 7. Non-additivity means that the representation of a phrase is a non-linear function of the words that constitute it. Because several experiments in this paper measure nonlinearity and attribution change under domain shift, switching to SCD would likely obscure the properties under observation, but might expose how interactions within non-additive phrases (as opposed to idioms or other multiword expressions) particularly evolve over time.

Other methods of feature credit attribution are deliberately hierarchical. The disadvantage of these alternatives is that they assume the very property of tree-based behavior that we are trying to discover. Singh et al. (2019a) hierarchically cluster sequentially proximate words based on CD, introducing **agglomerative contextual decomposition (ACD)**. Lundberg et al. (2019) similarly propose hierarchical clustering that is model-independent, based on any SHAP values.

5.1.2 Generalizing to Attentional Models

Do our results generalize to non-recurrent models? Perhaps not, but the methods can. After the work in this paper, we discovered that it fell into a recent line of work on **feature interaction**. In order to investigate the local behavior of constituents as they are learned or memorized, we can apply Shapley methods to any architecture, including fully attentional models. Where we can use SHAP, we can also develop interdependence metrics.

Some models already have related methods of deriving feature interaction metrics. In particular, Chen et al. (2020) brought the generic techniques of feature interaction (Lundberg and Lee, 2017; Lundberg et al., 2019) into Transformer models to test for syntactic structure. Because their methods already assume latent tree structure, we cannot test the presence of hierarchical behavior. However, by applying our method of normalizing by vector magnitude and potentially stratifying by sequential proximity, we could easily repeat these experiments or other theories of local hierarchical behavior on Transformer models as well.

Chapter 6

Beyond Words: The Development of Feature Sparsity

> "In this box are all the words I know," he said. "Most of them you will never need, some you will use constantly, but with them you may ask all the questions which have never been answered and answer all the questions which have never been asked."
>
> Norton Juster, *The Phantom Tollbooth*

In Chapters 4 and 5, we explored the temporal dynamics of models acquiring linguistic structure as training progresses. Here we turn to investigate more abstract properties of representations and how they evolve. In particular, the following work explores how **feature sparsity** naturally emerges in LMs during training.

Rather than consider the gradual development of hierarchical syntactic structure, we observe the dispersal or concentration (i.e., sparsity) of intermediate vector representations as a simple function of the frequency and predictability of each word. We better characterize the relationship between word frequency and the distribution of its "storage" in a model. The time dimension is essential to understanding the contributing factors to sparsity, because the model is more exposed to a word the longer it trains for, as well as in response to the word's corpus frequency.

As well as providing consideration of this crucial confounding factor between word frequency and a model's exposure to a word, viewing the time course of training again offers additional insights. We see how gradients responding to a word become sparser–meaning salient information is localized to a few neurons–as a network is exposed

more to that word. This validates work that considers individual neurons rather than entire subspaces (Torroba Hennigen et al., 2020; Durrani et al., 2020; Dalvi et al., 2019). We see that LSTM layers quickly correlate word frequency and sparsity, before the embedding layer surpasses the LSTMs in developing this correlation. The patterns we observe are, as in Chapter 4's discussion of the Information Bottleneck Hypothesis, possibly linked to a memorize/compress phase shift.

Publication Status This work was published in the Identifying and Understanding Deep Learning Phenomena Workshop at ICML 2019.

Sparsity Emerges Naturally in Neural Language Models

Naomi Saphra and **Adam Lopez**
n.saphra@ed.ac.uk alopez@ed.ac.uk
Institute for Language, Cognition, and Computation
University of Edinburgh

Abstract

Concerns about interpretability, computational resources, and principled inductive priors have motivated efforts to engineer sparse neural models for NLP tasks. If sparsity is important for NLP, might well-trained neural models naturally become roughly sparse? Using the Taxi-Euclidean norm to measure sparsity, we find that frequent input words are associated with concentrated or sparse activations, while frequent target words are associated with dispersed activations but concentrated gradients. We find that gradients associated with function words are more concentrated than the gradients of content words, even controlling for word frequency.

1 Introduction

Researchers in NLP have long relied on engineering features to reflect the sparse structures underlying language. Modern deep learning methods promised to relegate this practice to history, but have not eliminated the interest in sparse modeling for NLP. Along with concerns about computational resources (Chen et al., 2016; Narang et al., 2017b) and interpretability (Murphy et al., 2012; Subramanian et al., 2018), human intuitions continue to motivate sparse representations of language. For example, some work applies assumptions of sparsity to model latent hard categories such as syntactic dependencies (Padó and Lapata, 2007) or phonemes (Cotterell and Eisner, 2018). Niculae and Blondel (2017) found that a sparse attention mechanism outperformed dense methods on some NLP tasks; Narang et al. (2017a) found sparsified versions of LMs that outperform dense originals. Attempts to engineer sparsity rest on an unstated assumption that it doesn't arise naturally when neural models are learned. Is this true?

Using a simple measure of sparsity, we analyze how it arises in different layers of a neural language model in relation to word frequency. We show that the sparsity of a word representation increases with exposure to that word during training. We also find evidence of syntactic learning: gradient updates in backpropagation depend on whether a word's part of speech is open or closed class, even controlling for word frequency.

2 Methods

Language model. Our LM is trained on a corpus of tokenized, lowercased English Wikipedia (70/10/20 train/dev/test split). To reduce the number of unique words (mostly names) in the corpus, we excluded any sentence with a word which appears fewer than 100 times. Those words which still appear fewer than 100 times after this filter are replaced with <UNK>. The resulting training set is over 227 million tokens of around 19.5K types.

We use a standard 2-layer LSTM LM trained with cross entropy loss for 50 epochs. The pipeline from input x_{t-1} at time step $t-1$ to predicted output distribution $\hat{x}$ for time t is described in Figure 1, illustrating intermediate activations h_t^e, h_t^1, and h_t^2. At training time, the network observes x_t and backpropagates the gradient updates $\bar{h}_t^e$, $\bar{h}_t^1$, $\bar{h}_t^2$, and $\bar{x}_t$.

The embeddings produced by the encoding layer are 200 units, and the recurrent layers have 200 hidden units each. The batch size is set to forty, the maximum sequence length to 35, and the dropout ratio to 0.2. The optimizer is standard SGD with clipped gradients at $\ell_2 = 0.25$, where the learning rate begins at 20 and is quartered whenever loss fails to improve.

Measuring sparsity. We measure the sparsity of a vector v using the reciprocal of the Taxicab-Euclidean norm ratio (Repetti et al., 2015). This measurement has a long history as a measurement of sparsity in natural settings (Zibulevsky

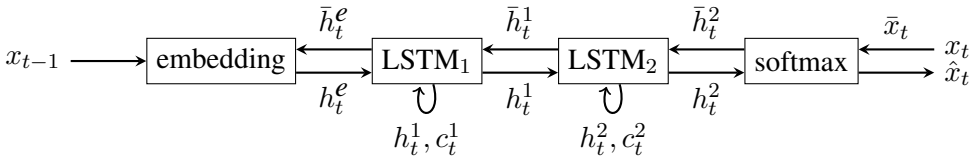


Figure 1: LM architecture for target word distribution $\hat{x}_t$, showing gradient updates from observed word x_t.

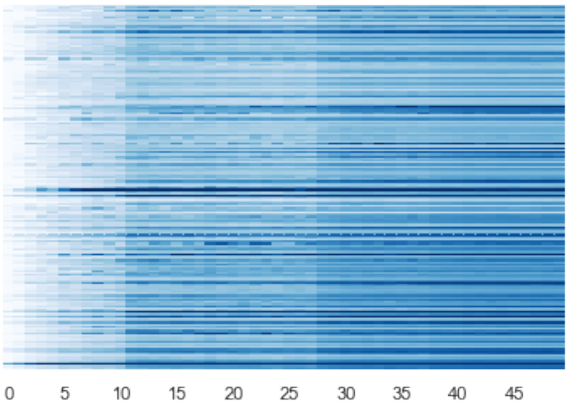
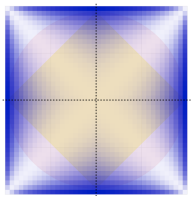


Figure 2: Average sparsity $\chi(\bar{h}_t^2)$ over all training epochs (x-axis), for target words x_t occurring more than 100k times in training. Target words are sorted from most frequent (bottom) to least frequent (top).

and Pearlmutter, 2001; Hoyer, 2004; Pham et al., 2017; Yin et al., 2014) and is formally defined as $\chi(v) = \|v\|_2/\|v\|_1$. The relationship between sparsity and this ratio is illustrated in two dimensions in the image on the right, in which darker blue regions are more concentrated. The pink circle shows the area where $\ell_2 \leq 1$ while the yellow diamond depicts $\ell_1 \leq 1$. For sparse vectors $\langle 1, 0\rangle$ or $\langle 0, 1\rangle$, the norms are identical so χ is 1, its maximum. For a uniform vector like $\langle 1, 1\rangle$, χ is at its smallest. In general, $\chi(v)$ is higher when most elements of v are close to 0; and lower when the elements are all similar in value.



3 Experiments

Sparsity is closely related to the behavior of a model: If only a few units hold most of the mass of a representation, the *activation* vector will be highly concentrated. If a neural network relies heavily on a small number of units in determining its predictions, the *gradient* will be highly concentrated. A highly concentrated gradient is mainly modifying a few specific pathways. For example, it might modify a neuron associated with particular inputs like parentheses (Karpathy et al., 2015), or properties like sentiment (Radford et al., 2017).

Representations of Target Words. Our first experiments look at the relationship of sparsity to target word x_t. Gradient updates triggered by the target are often used to identify units that are relevant to a prediction (Li et al., 2015), and as shown in Figure 2, gradient sparsity increases with both the frequency of a word in the corpus and the overall training time. In other words, more exposure leads to sparser relevance. Because the sparsity of $\bar{h}^2$ increases with target word frequency, we measure not sparsity itself but the Pearson correlation, over all words w, between word frequency and mean $\chi(h)$ over representations h where w is the target:

$$\rho_{\leftarrow}(h) = \text{corr}_w(\mu_{t:x_t=w}(\chi(h_t)), \text{freq}(w))$$

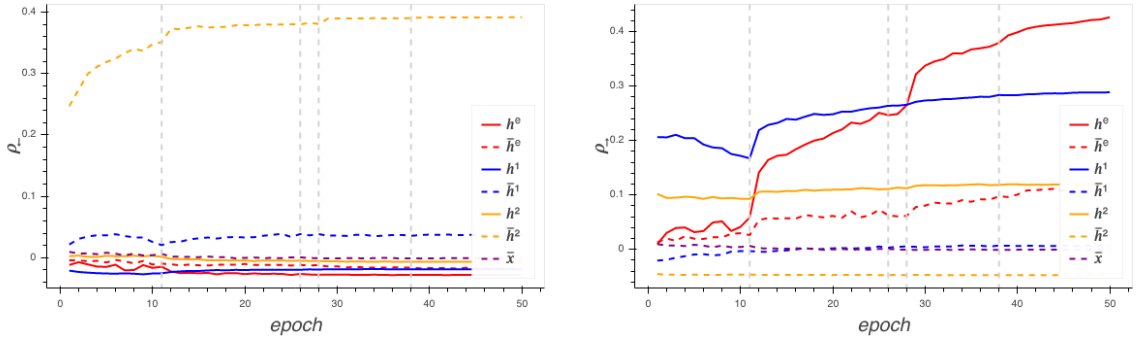
Here (Figure 3a) we confirm that concentrated gradients are not a result of concentrated activations, as activation sparsity $\chi(h^2)$ is not correlated with target word frequency.

The correlation is strong and increasing only for $\rho_{\leftarrow}(\bar{h}^2)$. The sparse structure being applied is therefore particular to the gradient passed from the softmax to the top LSTM layer, related to how a word interacts with its context.

The Role of Part of Speech. Figure 4 shows that $\rho_{\leftarrow}(\bar{h}^2)$ follows distinctly different trends for open POS classes[1] and closed classes[2]. To associate words to POS, we tagged our training corpus with spacy[3]; we associate a word to a POS only if the majority (at least 100) of its occurrences are tagged with that POS. We see that initially, frequent words from closed classes are highly concentrated, but soon stabilize, while frequent words from open classes continue to become more concentrated throughout training. Why?

Closed class words clearly signal POS. But open classes contain many ambiguous words, like

[1] ADJ, ADV, INTJ, NOUN, PROPN, VERB
[2] ADP, AUX, CCONJ, DET, PART, PRON, SCONJ
[3] https://spacy.io/



(a) $\rho_{\leftarrow}$ correlation with target word frequency

(b) $\rho_{\rightarrow}$ correlation with input word frequency

Figure 3: Correlation between mean sparsity of a word's representation and word frequency. Vertical dashed lines indicate when the optimizer has rescaled the step size.

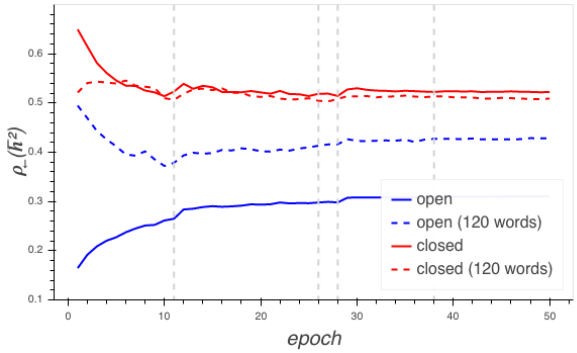


Figure 4: $\rho_{\leftarrow}(\bar{h}^2)$, evaluated over vocabulary from open and closed classes of POS.

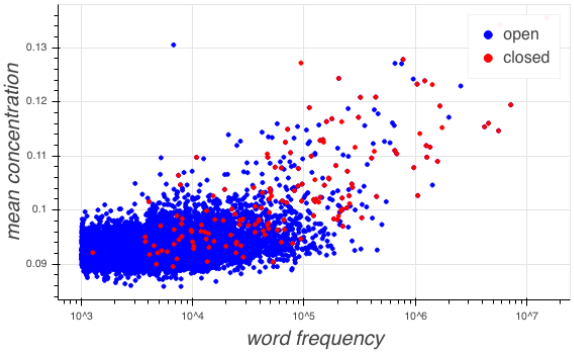


Figure 5: Mean sparsity of $\chi(\bar{h}^2)$ after 50 epochs, for words occurring more than 1k times in the train set.

"report", which can be a noun or verb. Open classes also contain many more words in general. We posit that early in training, closed classes reliably signal syntactic structure, and are essential for shaping network structure. But open classes are essential for predicting specific words, so their importance in training continues to increase after part of speech tags are effectively learned.

The high sparsity of function word gradient may be surprising when compared with findings that content words have a greater influence on outputs (Kádár et al., 2016). However, those findings were based on the impact on the vector representation of an entire sentence after omitting the word. Khandelwal et al. (2018) found that content words have a longer window during which they are relevant, which may explain the results of Kádár et al. (2016). Neither of these studies controlled for word frequency in their analyses contrasting content and function words, but we believe this oversight is alleviated in our work by measuring correlations rather than raw magnitude. Because $\rho_{\leftarrow}(\bar{h}^2)$ is higher when evaluated over more frequent words, which also tend to be function words (see Figure 5), we further control for the effect of frequency by including a measurement of trends in a sample of 120 words each from open and closed classes (Figure 4). This sample was selected by sorting all open and closed class words by frequency, then choosing a range of each sorted list with a similar average frequency.

Representations of Input Words. We next looked at the vector representations of each step in the word sequence as a representation of the input word x_{t-1} that produced that step. We measure the correlation with input word frequency:

$$\rho_{\rightarrow}(h) = \text{corr}_w(\mu_{t:x_{t-1}=w}(\chi(h_t)), \text{freq}(w))$$

Here (Figure 3b) we find that the view across training sheds some light on the learning process. While the lower recurrent layer quickly learns sparse representations of common input words, $\rho_{\rightarrow}(h^1)$ increases more slowly later in training and is eventually surpassed by $\rho_{\rightarrow}(h^{\text{e}})$, while gradient sparsity never becomes significantly correlated with word frequency. Li et al. (2016) studied the

activations of feedforward networks in terms of the importance of individual units by erasing a particular dimension and measuring the difference in log likelihood of the target class. They found that importance is concentrated into a small number of units at the lowest layers in a neural network, and is more dispersed at higher layers. Our findings suggest that this effect may be a natural result of the sparsity of the activations at lower layers.

We relate the trajectory over training to the Information Bottleneck Hypothesis of Shwartz-Ziv and Tishby (2017). This theory, connected to language model training by Saphra and Lopez (2018), proposes that the earlier stages of training are dedicated to learning to effectively represent inputs, while later in training these representations are compressed and the optimizer removes input information extraneous to the task of predicting outputs. If extraneous information is encoded in specific units, this compression would lead to the observed effect, in which the first time the optimizer rescales the step size, it begins an upward trend in $\rho_{\rightarrow}$ as extraneous units are mitigated.

4 Potential Explanations

Why do common target words have such concentrated gradients with respect to the final LSTM layer? A tempting explanation is that the amount of information we have about common words offers high confidence and stabilizes most of the weights, leading to generally smaller gradients. If this were true, the denominator of sparsity, gradient ℓ_1, should be strongly anti-correlated with word frequency. In fact, it is only ever slightly anti-correlated (correlation $> -.1$). Furthermore, the sparsity of the softmax gradient $\chi(\bar{x})$ does not exhibit the strong correlation seen in $\chi(\bar{h}^2)$, so sparsity at the LSTM gradient is not a direct effect of sparse logits.

However, the model could still be "high confidence" in terms of how it assigns blame for error during common events, even if it is barely more confident overall in its predictions. According to this hypothesis, a few specialized neurons might be responsible for the handling of such words.

Perhaps common words play a prototyping role that defines clusters of other words, and therefore have a larger impact on these clusters by acting as attractors within the representation space early on. Such a process would be similar to how humans acquire language by learning to use words like 'dog' before similar but less prototypical words like 'canine' (Rosch, 1999). As a possible mechanism for prototyping with individual units, Dalvi et al. (2019) found that some neurons in a translation system specialized in particular word forms, such as verb inflection or comparative and superlative adjectives. For example, a common comparative adjective like 'better' might be used as a reliable signal to shape the handling of comparatives by triggering specialized units, while rarer words have representations that are more distributed according to a small collection of specific contexts.

There may also be some other reason that common words interact more with specific substructures within the network. For example, it could be related to the use of context. Because rare words use more context than common words and content words use more context than function words (Khandelwal et al., 2018), the gradient associated with a common word would be focused on interactions with the most recent words. This would lead common word gradients to be more concentrated.

It is possible that frequent words have sparse activations because frequency is learned as a feature and thus is counted by a few dimensions of proportional magnitude, as posited by Li et al. (2016).

5 Potential Applications

Understanding where natural sparsity emerges in dense networks could be a useful guide in deciding which layers we can apply sparsity constraints to without affecting model performance, for the purpose of interpretability or efficiency. It might also explain why certain techniques are effective: for example, in some applications, summing representations together works quite well (Hill et al., 2016). We hypothesize that this occurs when the summed representations are sparse so there is often little overlap. Understanding sparsity could help identify cases where such simple ensembling approaches are likely to be effective.

Future work may develop ways of manipulating the training regime, as in curriculum learning, to accelerate the concentration of common words or incorporating concentration into the training objective as a regularizer. We would also like to see how sparsity emerges in models designed for specific end tasks, and to see whether concentration is a useful measure for the information compression predicted by the Information Bottleneck.

References

Yunchuan Chen, Lili Mou, Yan Xu, Ge Li, and Zhi Jin. 2016. Compressing neural language models by sparse word representations. *arXiv preprint arXiv:1610.03950*.

Ryan Cotterell and Jason Eisner. 2018. A deep generative model of vowel formant typology. *arXiv preprint arXiv:1807.02745*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Patrik O Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and Understanding Recurrent Networks. *arXiv:1506.02078 [cs]*. ArXiv: 1506.02078.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. *arXiv:1805.04623 [cs]*. ArXiv: 1805.04623.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *arXiv:1602.08952 [cs]*. ArXiv: 1602.08952.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in NLP. *arXiv preprint arXiv:1506.01066*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. *Proceedings of COLING 2012*, pages 1933–1950.

Sharan Narang, Gregory Diamos, Shubho Sengupta, and Erich Elsen. 2017a. Exploring Sparsity in Recurrent Neural Networks. *arXiv:1704.05119 [cs]*. ArXiv: 1704.05119.

Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. 2017b. Exploring sparsity in recurrent neural networks. *arXiv preprint arXiv:1704.05119*.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems*, pages 3338–3348.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Mai Quyen Pham, Benoit Oudompheng, Jérôme I Mars, and Barbara Nicolas. 2017. A noise-robust method with smoothed 1/2 regularization for sparse moving-source mapping. *Signal Processing*, 135:96–106.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to Generate Reviews and Discovering Sentiment. *arXiv:1704.01444 [cs]*. ArXiv: 1704.01444.

Audrey Repetti, Mai Quyen Pham, Laurent Duval, Emilie Chouzenoux, and Jean-Christophe Pesquet. 2015. Euclid in a taxicab: Sparse blind deconvolution with smoothed ℓ_1/ℓ_2 regularization. *IEEE Signal Processing Letters*, 22(5):539–543.

Eleanor Rosch. 1999. Principles of categorization. *Concepts: core readings*, 189.

Naomi Saphra and Adam Lopez. 2018. Understanding learning dynamics of language models with svcca. *arXiv preprint arXiv:1811.00225*.

Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the Black Box of Deep Neural Networks via Information. *arXiv:1703.00810 [cs]*. ArXiv: 1703.00810.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Penghang Yin, Ernie Esser, and Jack Xin. 2014. Ratio and difference of l1 and l2 norms and sparse representation with coherent dictionaries. *Commun. Inform. Systems*, 14(2):87–109.

Michael Zibulevsky and Barak A Pearlmutter. 2001. Blind source separation by sparse decomposition in a signal dictionary. *Neural computation*, 13(4):863–882.

6.1 Comments on the Paper

In this paper, we have investigated the sparsity of activations and, using the same sparsity metric applied to gradients, observed how localized change is across neurons.

6.1.1 Localization of Changes

The local loss surface revealed by gradients is not the only way of viewing change across neurons. Because the optimizer does not follow this surface perfectly, and the training data is not the only way of viewing change in error, we may want to investigate change in each neuron through **Loss Change Allocation** (LCA) (Lan et al., 2019).

6.1.1.1 Loss Change Allocation

LCA is a method that does not look directly at the gradient with respect to training error, but instead allocates responsibility for *actual* error change after a training step.

If we consider the loss function over a training set $L(\theta_t)$ with the parameters at timestep t, LCA is based on a first order Taylor approximation of the path between two timesteps:

$$L(\theta_{t+1}) - L(\theta_t) = \int_{\theta_t}^{\theta_{t+1}} < \nabla_\theta L(\theta_t), d\theta > \tag{6.1}$$

$$\approx < \nabla_\theta L(\theta_t), \theta_{t+1} - \theta_t > \tag{6.2}$$

This dot product can be reformulated into an element-wise operation, such that we can consider the individual parameters that contribute to this change in loss. We therefore define the LCA allocated to parameter unit $\theta^{(i)}$ like so:

$$\sum_{i=0}^{K-1} A_{t,i} = \sum_{i=0}^{K-1} (\nabla_\theta L(\theta_t))^{(i)} (\theta_{t+1}^{(i)} - \theta_t^{(i)}) \tag{6.3}$$

This way, we could consider whether following the gradient in a particular direction actually leads to a particular unit improving or damaging performance when considered in combination with the other changes made during that timestep.

Using LCA, Lan et al. (2019) found that only a little over half of parameters actually help improve performance in their changes at any given time step, with some entire layers moving against the gradient in a particular timestep and damaging performance. Because all layers tend to move in synchrony, they hypothesize that these damaging layers lag behind others while they oscillate back and forth over valleys in the optimization landscape, therefore moving against the overall gradient motion. These dynamics

calls into question a simplistic view of the training process through the lens of the gradient landscape itself. Even from timestep to timestep, the movement of parameters along the gradient does not translate directly to improvement in the training objective.

6.1.2 Generalizing to Attentional Models

Current Transformer models provide new motivations and methods for considering sparsity during the forward pass through a neural network. Attention distributions can be easily measured in terms of their entropy, which offers additional information about how localized word importance itself might be, as it gives a view of how sparse *attention* naturally is across words.

Chapter 7

Conclusion

> I don't know of any wrong road to Dictionopolis, so if this road goes to Dictionopolis at all it must be the right road, and if it doesn't it must be the right road to somewhere else, because there are no wrong roads to anywhere.
>
> Norton Juster, *The Phantom Tollbooth*

It is surely unreasonable to claim that a machine achieves high accuracy on a language task by using the same strategies as a human[1]. However, in order to generalize well in the way humans do, we might assume that it is useful for a model to tend to pick up on similar linguistic structure and cues. Human strategies, often assumed to be encoded in our brains from birth, are a useful bias to have when handling human languages.

Even when a neural network apparently adopts the language habits we expect from a human strategy, it does not necessarily learn as a human does. The ways in which it matches or diverges from human learning give useful hints as to how it matches or diverges from human language mechanisms and innate biases. If our findings show anything, they illustrate that an LSTM should not be analyzed like a human brain; it should be understood as an artificial neural network.

Our methods point to several possible indicators of memorization/compression phase shifts (Chapters 4 and 6), most evident in embedding dictionaries. We have also discovered that distant content information is removed early in training (Chapter 4), pointing to the essential role of gradually building long distance relations. We even directly

[1] Or as Fred Jelinek famously put it: "Every time I fire a linguist, the performance goes up."

witness construction of these distant dependencies, explaining how compositional behavior emerges, by testing in synthetic environments (Chapter 5). These findings are just a few of the results from our new methodologies, which are designed to study the timecourse of neural network training.

7.1 The growing field of NLP training dynamics

When the work in this thesis was first published, it constituted a proposal to connect the new science of neural network training dynamics with the nascent trend of relating contextual word representations to linguistic properties. Now, questions about how word representations develop during training are becoming commonplace.

In 2020, the year following the publication of Saphra and Lopez (2019), the first of the papers in this thesis, the NLP community began to take a serious look at what happens during the LM training process. Chiang et al. (2020) performed "embryology" on the pretrained attentional model ALBERT (Lan et al., 2020), finding that it learned to predict different parts of speech at different speeds. Liu et al. (2021) further found that RoBERTa acquired different probed properties at different rates. Hao et al. (2020) and Merchant et al. (2020) investigated learning dynamics during fine-tuning of attentional models, confirming through a variety of techniques that most changes occur in the last layer during the task change. Zhang et al. (2020) and Warstadt et al. (2020) both measured performance of RoBERTa (Liu et al., 2019b) pretraining when exposed to varying corpus sizes, an important conflation factor for exposure to more data due to increased training time.

Hao et al. (2020) and Merchant et al. (2020) represent a particular new subgenre of scientific work in understanding LM training: what we might call **fine-tuning research**[2], studying how self-supervised pretraining positions a model to be efficiently fine-tuned for a supervised task. Dodge et al. (2020), for example, revealed that fine-tuning seeds with better accuracy immediately after pretraining tend to produce fine-tuned models with better accuracy after training. Pretrained representations are even highly robust to pruning immediately before finetuning, as found by both Prasanna et al. (2020) and Radiya-Dixit and Wang (2020). However, a dearth of access to checkpoints (at least of massive modern models) during pretraining limits similar research of training dynamics *prior* to the start of finetuning.

[2]Though Wang (2020) calls it **developmental BERTology**.

The taste for a clear understanding of how linguistic properties evolve is perhaps best exemplified by the release of LINSPECTOR WEB (Eichler et al., 2019), a probing suite for word representations which handles multiple languages. This tool was released with a feature that allows it to probe multiple checkpoints during training, with the declaration that it was "considered a crucial feature as it provides insights on learning dynamics of models (Saphra and Lopez, 2019)".

7.2 Future work

The work presented in this thesis is only the very beginning of the possibilities in NLP training dynamics. Most obviously, future work should expand into modern fully attentional networks. Beyond that, a number of goals and methods have been left yet untouched.

7.2.1 Loss landscapes

None of the work in this thesis has explicitly considered optimization processes or the loss landscape being explored. This is a major gap, given that such methods have led to a number of insights.

One property of interest is whether you can shift the final weights from their learned space slightly and maintain similar performance, or whether instead performance abruptly degrades even for very close alternative parameter settings. There is a long-standing debate about whether this property, the **width** of the optimum, determines whether a solution generalizes to a test set (Li et al., 2018; Dinh et al., 2017; Keskar et al., 2017; Hochreiter and Schmidhuber, 1997a). Some research has suggested that pretrained representations tend to be at wider optima (Hao et al., 2019), which may hint at why pretraining is such an effective strategy.

Other landscape properties are yet unexplored, even at the pretrain/finetune boundary. **Mode connectivity** (Garipov et al., 2018; Draxler et al., 2018; Frankle et al., 2020) yields the surprising insight that overparameterized models like modern DNNs have landscapes of massive valleys, in which every point with a particular training loss is connected to every other such point, and have a connecting path where the loss remains the same through the entire path. Empirically, in simple vision contexts, the *test loss* even remains the same along the whole path. **Module criticality** is predicted

by whether there is a wide valley connecting initial and final values of a module, and is predictive of generalization in the final model (Chatterji et al., 2020). These concepts are largely unexplored for the pretraining process itself, outside of finetuning research.

7.2.2 Visualization

Techniques like Multi-Dimensional Scaling (MDS; as used by Saxe et al. (2019) and shown in Figure 3.4) and M-PHATE (Gigante et al., 2019) have provided 2-dimensional visualizations that are particular to the training process. These methods ensure that local movement from epoch to epoch remains clearly connected in the resulting visualization. What if we looked at specific words, or particular inflections, and how they relate to each other over the course of training? What insights might these methods yield?

7.2.3 Understanding phases and early training

Section 3.1 described a number of phase transition phenomena that have been observed in neural networks. However, we don't understand how they relate to each other well. Could a detailed study of different phase transitions lead to better understanding of the early stages of training? When it comes to language, are these phases synchronized or do they apply to different concepts individually, as different POS tags are learned at different speeds (Chiang et al., 2020)?

7.2.4 Trajectory analysis

Much recent work in training dynamics has focused on the **Neural Tangent Kernel** (NTK; Jacot et al., 2018). In this formulation, we approximate the gradient trajectory with a linear function, resulting in an approximation that is empirically close to the true gradient trajectory. A linear function is easy to employ in proof work, compared to most gradients—but usually, this technique requires unrealistic assumptions like an infinite-width neural network, and it has been used mostly on simplistic architectures with toy problems. Hu et al. (2020), however, use the NTK of a network early in training to study dynamics in a finite-width setting, so we are likely to see more trajectory results that apply to less constrained settings. Even without NTK methods, Lu et al. (2021) proved that attention key-value weights mutually reinforce each other's magnitude during training. Trajectory-based methods, NTK or otherwise, may be one way to

study how inductive bias helps learn latent tree structure or other underlying semantics.

7.2.5 Improving future models

Does this help us improve future models? First we would have to answer whether magnifying a natural bias strengthens or weakens a model. For example, if a model's attention doesn't actually resemble human attention in reading, then there is no reason to believe that it could be improved by making its development more humanlike, such as by imposing heuristics based on sentence complexity. Models clearly learn with an implicit curriculum (Wu et al., 2020b), but it is hard to find high-resource tasks where a curriculum ordered based on easiness improves on the default training stages. However, the results in Chapter 5 suggest a possible reason why a curriculum for learning short or syntactically shallow sequences first is unproductive: these examples are already used as scaffolding for longer dependencies.

Other insights might point us towards enhancing the current models or even developing better ones in the future. Chapter 6 tells us that some layers produce naturally sparse representations; with hardware that takes advantage of sparsity, we can make some weights and representations completely sparse based on these insights, while allowing others to remain dense. However, even if insights about training dynamics never improve the performance of models in practice, understanding why systems work is basic scientific research. Modern models might be far more trusted if they were better understood.

Training dynamics as a field continues to develop new techniques for analyzing the learning process, while the NLP interpretability community continues to yield new discoveries about the inner workings of language models. The work of understanding how language develops in a neural network is far from complete, but the intersection of these two communities is growing. Our understanding of language model development is still expanding, moving into an increasingly complex description of the world.

Bibliography

Abnar, S., Dehghani, M., and Zuidema, W. (2020). Transferring Inductive Biases through Knowledge Distillation. *arXiv:2006.00555 [cs, stat]*. arXiv: 2006.00555.

Achille, A., Rovere, M., and Soatto, S. (2019). Critical learning periods in deep networks. In *International Conference on Learning Representations*.

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2017). Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In *International Conference on Learning Representations*.

Alain, G. and Bengio, Y. (2018). Understanding intermediate layers using linear classifier probes. *arXiv:1610.01644 [cs, stat]*. arXiv: 1610.01644.

Andersen, H. (1989). Markedness theory—the first 150 years. *Markedness in synchrony and diachrony*, 46.

Arora, S., Cohen, N., Hu, W., and Luo, Y. (2019). Implicit Regularization in Deep Matrix Factorization.

Arpit, D., Jastrzundefinedbski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. (2017). A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 233–242. JMLR.org.

Arras, L., Horn, F., Montavon, G., Müller, K.-R., and Samek, W. (2016). Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany. Association for Computational Linguistics.

Bastings, J. and Filippova, K. (2020). The elephant in the interpretability room:

Why use attention as explanation when we have saliency methods? *ArXiv*, abs/2010.05607.

Bau, A., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2019). Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., and Glass, J. (2017). What do Neural Machine Translation Models Learn about Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Belinkov, Y. and Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Bhattamishra, S., Ahuja, K., and Goyal, N. (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.

Bock, J. K. (1986). Syntactic persistence in language production. *Cognitive Psychology*, 18(3):355–387.

Bogoychev, N., Heafield, K., Aji, A. F., and Junczys-Dowmunt, M. (2018). Accelerating asynchronous stochastic gradient descent for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2991–2996, Brussels, Belgium. Association for Computational Linguistics.

Bowman, S. R., Manning, C. D., and Potts, C. (2015). Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches - Volume 1583*, COCO'15, page 37–42, Aachen, DEU. CEUR-WS.org.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G. M., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., Mc-

Candlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. In *Proceedings of NeurIPS 2020*.

Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., and Wattenhofer, R. (2020). On identifiability in transformers. In *International Conference on Learning Representations*.

Chatterji, N., Neyshabur, B., and Sedghi, H. (2020). The intriguing role of module criticality in the generalization of deep networks. In *International Conference on Learning Representations*.

Chen, H., Zheng, G., and Ji, Y. (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.

Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. (2019). L-shapley and c-shapley: Efficient model interpretation for structured data. In *International Conference on Learning Representations*.

Cheng, H., Lian, D., Gao, S., and Geng, Y. (2018). Evaluating capability of deep neural networks for image classification via information plane. In *ECCV*.

Chiang, C.-H., Huang, S.-F., and Lee, H.-y. (2020). Pretrained language model embryology: The birth of ALBERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6813–6828, Online. Association for Computational Linguistics.

Chomsky, N. and Keyser, S. J. (1988). *Language and problems of knowledge: The Managua lectures*, volume 16. MIT press.

Chrupala, G. (2019). Symbolic Inductive Bias for Visually Grounded Learning of Spoken Language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6452–6462, Florence, Italy. Association for Computational Linguistics.

Chrupala, G. and Alishahi, A. (2019). Correlating Neural and Symbolic Representations of Language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2952–2962, Florence, Italy. Association for Computational Linguistics.

Chung, Y.-A., Belinkov, Y., and Glass, J. (2020). Similarity Analysis of Self-Supervised Speech Representations. *arXiv:2010.11481 [cs, eess]*. arXiv: 2010.11481.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019). What Does BERT Look At? An Analysis of BERT's Attention. *BlackboxNLP*.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. (2018). What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Corkery, M., Matusevych, Y., and Goldwater, S. (2019). Are we there yet? encoder-decoder neural networks as cognitive models of English past tense inflection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3868–3877, Florence, Italy. Association for Computational Linguistics.

Cresswell, M. (1973). *Logics and languages*. Routledge.

Dalvi, F., Durrani, N., Sajjad, H., Belinkov, Y., Bau, A., and Glass, J. (2019). What Is One Grain of Sand in the Desert? Analyzing Individual Neurons in Deep NLP Models. In *AAAI Conference on Artificial Intelligence*. arXiv: 1812.09355.

Darwin, C. (1859). *On the origin of species: A facsimile of the first edition*. John Murray.

Davies, N. (2015). *Cuckoo: Cheating by Nature*. Bloomsbury.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. (2017). Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1019–1028. JMLR.org.

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., and Smith, N. (2020). Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv:2002.06305 [cs]*. arXiv: 2002.06305.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. (2018). Essentially no

barriers in neural network energy landscape. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318. PMLR.

Durrani, N., Sajjad, H., Dalvi, F., and Belinkov, Y. (2020). Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4865–4880, Online. Association for Computational Linguistics.

Eichler, M., Şahin, G. G., and Gurevych, I. (2019). LINSPECTOR WEB: A Multilingual Probing Suite for Word Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 127–132, Hong Kong, China. Association for Computational Linguistics.

Elazar, Y., Ravfogel, S., Jacovi, A., and Goldberg, Y. (2020). Amnesic Probing: Behavioral Explanation with Amnesic Counterfactuals. *TACL*. arXiv: 2006.00995.

Emelin, D., Titov, I., and Sennrich, R. (2020). Detecting word sense disambiguation biases in machine translation for model-agnostic adversarial attacks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7635–7653, Online. Association for Computational Linguistics.

Ethayarajh, K. (2019). How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Feng, S., Wallace, E., Grissom II, A., Iyyer, M., Rodriguez, P., and Boyd-Graber, J. (2018). Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. (2020). Mode connectivity and sparse neural networks.

Frazier, L. and Fodor, J. D. (1978). The sausage machine: A new two-stage parsing model. *Cognition*, 6(4):291–325.

Futrell, R. and Levy, R. (2017). Noisy-context surprisal as a human sentence processing cost model. In *Proceedings of the 15th conference of the european chapter of the association for computational linguistics: Volume 1, long papers*, pages 688–698.

Futrell, R., Wilcox, E., Morita, T., Qian, P., Ballesteros, M., and Levy, R. (2019). Neural language models as psycholinguistic subjects: Representations of syntactic state. 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 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Ghorbani, A. and Zou, J. Y. (2020). Neuron shapley: Discovering the responsible neurons. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5922–5932. Curran Associates, Inc.

Gigante, S., Charles, A. S., Krishnaswamy, S., and Mishne, G. (2019). Visualizing the phate of neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Giulianelli, M., Harding, J., Mohnert, F., Hupkes, D., and Zuidema, W. (2018). Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Godard, P., Zanon Boito, M., Ondel, L., Berard, A., Yvon, F., Villavicencio, A., and Besacier, L. (2018). Unsupervised Word Segmentation from Speech with Attention. In *Interspeech 2018*, Hyderabad, India.

Goldfeld, Z., Van Den Berg, E., Greenewald, K., Melnyk, I., Nguyen, N., Kingsbury, B., and Polyanskiy, Y. (2019). Estimating information flow in deep neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2299–2308. PMLR.

Goodfellow, I. J., Vinyals, O., and Saxe, A. M. (2015). Qualitatively characterizing neural network optimization problems. *arXiv:1412.6544 [cs, stat]*. arXiv: 1412.6544.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (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.

Hahn, M. (2020). Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Hall Maudslay, R., Valvoda, J., Pimentel, T., Williams, A., and Cotterell, R. (2020). A tale of a probe and a parser. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7389–7395, Online. Association for Computational Linguistics.

Hao, Y., Dong, L., Wei, F., and Xu, K. (2019). Visualizing and understanding the effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4143–4152, Hong Kong, China. Association for Computational Linguistics.

Hao, Y., Dong, L., Wei, F., and Xu, K. (2020). Investigating Learning Dynamics of BERT Fine-Tuning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 87–92, Suzhou, China. Association for Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Herbelot, A. (2020). How to Stop Worrying About Compositionality. *The Gradient*.

Hewitt, J., Hahn, M., Ganguli, S., Liang, P., and Manning, C. D. (2020). RNNs can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1978–2010, Online. Association for Computational Linguistics.

Hewitt, J. and Liang, P. (2019). Designing and Interpreting Probes with Control Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Hewitt, J. and Manning, C. D. (2019). A Structural Probe for Finding Syntax in Word Representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Hirsch, H. V. and Spinelli, D. (1971). Modification of the distribution of receptive field orientation in cats by selective visual exposure during development. *Experimental brain research*, 12(5):509–527.

Hochreiter, S. and Schmidhuber, J. (1997a). Flat minima. *Neural Computation*, 9:1–42.

Hochreiter, S. and Schmidhuber, J. (1997b). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Hsu, T., Liu, C., and Lee, H. (2019). Zero-shot reading comprehension by cross-lingual transfer learning with multi-lingual language representation model. *CoRR*, abs/1909.09587.

Htut, P. M., Phang, J., Bordia, S., and Bowman, S. R. (2019). Do Attention Heads in BERT Track Syntactic Dependencies? *arXiv:1911.12246 [cs]*. arXiv: 1911.12246.

Hu, W., Xiao, L., Adlam, B., and Pennington, J. (2020). The Surprising Simplicity of the Early-Time Learning Dynamics of Neural Networks. *arXiv:2006.14599 [cs, stat]*. arXiv: 2006.14599.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2020). Compositionality decomposed: How do neural networks generalise? *J. Artif. Intell. Res.*, 67:757–795.

Hupkes, D. and Zuidema, W. H. (2018). Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *ArXiv*, abs/1711.10203.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jain, S. and Wallace, B. C. (2019). Attention is not Explanation. 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 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho*, K., and Geras*, K. (2020). The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*.

Jin, X., Wei, Z., Du, J., Xue, X., and Ren, X. (2020). Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *International Conference on Learning Representations*.

Jumelet, J., Zuidema, W., and Hupkes, D. (2019). Analysing neural language models: Contextual decomposition reveals default reasoning in number and gender assignment. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 1–11, Hong Kong, China. Association for Computational Linguistics.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing, Second Edition*. MIT Press, Cambridge, MA.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray,

S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv:2001.08361 [cs, stat]*. arXiv: 2001.08361.

Karpathy, A., Johnson, J., and Fei-Fei, L. (2015). Visualizing and Understanding Recurrent Networks. *arXiv:1506.02078 [cs]*. arXiv: 1506.02078.

Keskar, N., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kirov, C. and Cotterell, R. (2018). Recurrent neural networks in linguistic theory: Revisiting pinker and prince (1988) and the past tense debate. *Transactions of the Association for Computational Linguistics*, 6:651–665.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of Neural Network Representations Revisited. *arXiv preprint arXiv:1905.00414*.

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2. Publisher: Frontiers.

Kuncoro, A., Dyer, C., Hale, J., Yogatama, D., Clark, S., and Blunsom, P. (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)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Lakretz, Y., Kruszewski, G., Desbordes, T., Hupkes, D., Dehaene, S., and Baroni, M. (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.

Lan, J., Liu, R., Zhou, H., and Yosinski, J. (2019). LCA: Loss Change Allocation for Neural Network Training. *arXiv:1909.01440 [cs, stat]*. arXiv: 1909.01440.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020).

Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Lanorte, A., Lasaponara, R., Lovallo, M., and Telesca, L. (2014). Fisher-shannon information plane analysis of spot/vegetation normalized difference vegetation index (ndvi) time series to characterize vegetation recovery after fire disturbance. *Int. J. Appl. Earth Obs. Geoinformation*, 26:441–446.

LeCun, Y. and Cortes, C. (2005). The mnist database of handwritten digits.

Lee, C. M., Liu, J., and Peng, W. (2020). Applying cyclical learning rate to neural machine translation. *arXiv preprint arXiv:2004.02401*.

Lepori, M. and McCoy, R. T. (2020). Picking BERT's brain: Probing for linguistic dependencies in contextualized embeddings using representational similarity analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3637–3651, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Levy, O. and Goldberg, Y. (2014). Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems*.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the Loss Landscape of Neural Nets. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc.

Li, J., Chen, X., Hovy, E., and Jurafsky, D. (2015). Visualizing and understanding neural models in NLP. *arXiv preprint arXiv:1506.01066*.

Li, J., Monroe, W., and Jurafsky, D. (2016). Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Limisiewicz, T. and Marecek, D. (2020). Syntax Representation in Word Embeddings and Neural Networks–A Survey. *arXiv preprint arXiv:2010.01063*.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Liu, L. Z., Wang, Y., Kasai, J., Hajishirzi, H., and Smith, N. A. (2021). Probing

Across Time: What Does RoBERTa Know and When? *arXiv:2104.07885 [cs]*. arXiv: 2104.07885.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019a). Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lu, H., Mao, Y., and Nayak, A. (2021). On the dynamics of training attention models. In *International Conference on Learning Representations*.

Lu, K., Mardziel, P., Leino, K., Fedrikson, M., and Datta, A. (2020). Influence Paths for Characterizing Subject-Verb Number Agreement in LSTM Language Models. *arXiv:2005.01190 [cs]*. arXiv: 2005.01190.

Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *NIPS*.

Lundberg, S. M., Erion, G. G., and Lee, S.-I. (2019). Consistent Individualized Feature Attribution for Tree Ensembles. *arXiv:1802.03888 [cs, stat]*. arXiv: 1802.03888.

MacDonald, M. C., Pearlmutter, N. J., and Seidenberg, M. S. (1994). The lexical nature of syntactic ambiguity resolution. *Psychological review*, 101(4):676.

Mangalam, K. and Prabhu, V. U. (2019). Do deep neural networks learn shallow learnable examples first? In *ICML Deep Phenomena Workshop*.

Marecek, D. and Rosa, R. (2019). From Balustrades to Pierre Vinken: Looking for Syntax in Transformer Self-Attentions. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 263–275, Florence, Italy. Association for Computational Linguistics.

McCoy, R. T., Frank, R., and Linzen, T. (2020). Does syntax need to grow on trees? sources of hierarchical inductive bias in sequence-to-sequence networks. *Transactions of the Association for Computational Linguistics*, 8:125–140.

Merchant, A., Rahimtoroghi, E., Pavlick, E., and Tenney, I. (2020). What Happens To BERT Embeddings During Fine-tuning? *arXiv:2004.14448 [cs]*. arXiv: 2004.14448.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Merrill, W., Weiss, G., Goldberg, Y., Schwartz, R., Smith, N. A., and Yahav, E. (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.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Miller, G. A. and Chomsky, N. (1963). Finitary models of language users.

Mimno, D. and Thompson, L. (2017). The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2873–2878, Copenhagen, Denmark. Association for Computational Linguistics.

Morcos, A., Raghu, M., and Bengio, S. (2018a). Insights on representational similarity in neural networks with canonical correlation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Morcos, A. S., Barrett, D. G., Rabinowitz, N. C., and Botvinick, M. (2018b). On the importance of single directions for generalization. In *International Conference on Learning Representations*.

Morerio, P., Cavazza, J., Volpi, R., Vidal, R., and Murino, V. (2017). Curriculum dropout. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3564–3572.

Movva, R. and Zhao, J. (2020). Dissecting lottery ticket transformers: Structural and behavioral study of sparse neural machine translation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 193–203, Online. Association for Computational Linguistics.

Mu, J. and Viswanath, P. (2018). All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Murdoch, W. J., Liu, P. J., and Yu, B. (2018). Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs. In *ICLR*.

Noshad, M., Zeng, Y., and Hero, A. O. (2019). Scalable mutual information estimation using dependence graphs. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2962–2966.

Papadimitriou, I. and Jurafsky, D. (2020). Learning Music Helps You Read: Using Transfer to Study Linguistic Structure in Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6829–6839, Online. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Pimentel, T., Saphra, N., Williams, A., and Cotterell, R. (2020a). Pareto Probing: Trading Off Accuracy for Complexity. In *EMNLP*. arXiv: 2010.02180.

Pimentel, T., Valvoda, J., Hall Maudslay, R., Zmigrod, R., Williams, A., and Cotterell, R. (2020b). Information-Theoretic Probing for Linguistic Structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Plunkett, K. and Marchman, V. (1991). U-shaped learning and frequency effects in a multi-layered perception: Implications for child language acquisition. *Cognition*, 38:43–102.

Popel, M. and Bojar, O. (2018). Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Prasanna, S., Rogers, A., and Rumshisky, A. (2020). When BERT Plays the Lottery,

All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229, Online. Association for Computational Linguistics.

Quiller-Crouch, A. (1916). *On the Art of Writing*. Read Books.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Radiya-Dixit, E. and Wang, X. (2020). How fine can fine-tuning be? learning efficient language models. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2435–2443. PMLR.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017). Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ravfogel, S., Goldberg, Y., and Linzen, T. (2019). Studying the inductive biases of RNNs with synthetic variations of natural languages. 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 3532–3542, Minneapolis, Minnesota. Association for Computational Linguistics.

Ravichander, A., Belinkov, Y., and Hovy, E. (2021). Probing the probing paradigm: Does probing accuracy entail task relevance? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3363–3377, Online. Association for Computational Linguistics.

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., and Kim, B. (2019). Visualizing and Measuring the Geometry of BERT. In *Proceedings of Neural Information Processing Systems*, pages 8594–8603.

Reiter, E. (2007). Last Words: The Shrinking Horizons of Computational Linguistics. *Computational Linguistics*, 33(2):283–287.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What

we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Rogers, T. and McClelland, J. L. (2004). *Semantic Cognition: A Parallel Distributed Processing Approach*. A Bradford Book.

Rosenfeld, R. (2000). Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Rudinger, R., Naradowsky, J., Leonard, B., and Durme, B. V. (2018). Gender bias in coreference resolution. In *NAACL-HLT*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Saphra, N. and Lopez, A. (2016). Evaluating Informal-Domain Word Representations With UrbanDictionary. In *The First Workshop on Evaluating Vector Space Representations for NLP*.

Saphra, N. and Lopez, A. (2019). Understanding Learning Dynamics Of Language Models with SVCCA. In *NAACL*. arXiv: 1811.00225.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. (2018). On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (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.

Schneider, N. and Smith, N. A. (2015). A corpus and model integrating multiword expressions and supersenses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1537–1547, Denver, Colorado. Association for Computational Linguistics.

Serrano, S. and Smith, N. A. (2019). Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Shannon, C. E. (1948). A Mathematical Theory of Communication.

Shapley, L. S. (1953). Stochastic Games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100. Publisher: National Academy of Sciences Section: Mathematics.

Shwartz-Ziv, R. and Tishby, N. (2017). Opening the Black Box of Deep Neural Networks via Information. *arXiv:1703.00810 [cs]*. arXiv: 1703.00810.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Singh, C., Murdoch, W. J., and Yu, B. (2019a). Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*.

Singh, J., McCann, B., Socher, R., and Xiong, C. (2019b). BERT is Not an Interlingua and the Bias of Tokenization. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 47–55, Hong Kong, China. Association for Computational Linguistics.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. (2020). Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv:physics/0004057*. arXiv: physics/0004057.

Tishby, N. and Zaslavsky, N. (2015). Deep Learning and the Information Bottleneck Principle. *arXiv:1503.02406 [cs]*. arXiv: 1503.02406.

Torroba Hennigen, L., Williams, A., and Cotterell, R. (2020). Intrinsic probing through dimension selection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 197–216, Online. Association for Computational Linguistics.

Tran, K., Bisazza, A., and Monz, C. (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.

Vashishth, S., Upadhyay, S., Tomar, G. S., and Faruqui, M. (2019). Attention Interpretability Across NLP Tasks. *arXiv:1909.11218 [cs]*. arXiv: 1909.11218.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Vedaldi, A., Mahendran, S., Tsogkas, S., Maji, S., Girshick, B., Kannala, J., Rahtu, E., Kokkinos, I., Blaschko, M. B., Weiss, D., Taskar, B., Simonyan, K., Saphra, N., and Mohamed, S. (2014). Understanding Objects in Detail with Fine-grained Attributes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Vig, J. and Belinkov, Y. (2019). Analyzing the Structure of Attention in a Transformer Language Model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Voita, E., Sennrich, R., and Titov, I. (2019a). The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019b). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Voita, E. and Titov, I. (2020). Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Wang, X. (2020). The Curious Case of Developmental BERTology. *TOPBOTS*.

Wanner, E. (1980). The ATN and the sausage machine: Which one is baloney? *Cognition*. Publisher: Elsevier Science.

Warstadt, A., Zhang, Y., Li, X., Liu, H., and Bowman, S. R. (2020). Learning Which Features Matter: RoBERTa Acquires a Preference for Linguistic Generalizations (Eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, Online. Association for Computational Linguistics.

Weiss, G., Goldberg, Y., and Yahav, E. (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.

Wickstrøm, K., Løkse, S., Kampffmeyer, M., Yu, S., Príncipe, J., and Jenssen, R. (2019). Information plane analysis of deep neural networks via matrix-based renyi's entropy and tensor kernels. *ArXiv*, abs/1909.11396.

Wiegreffe, S. and Pinter, Y. (2019). Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Wiesel, T. N. and Hubel, D. H. (1963). Effects of visual deprivation on morphology and physiology of cells in the cat's lateral geniculate body. *Journal of neurophysiology*, 26(6):978–993.

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.*, 1:67–82.

Wu, J., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2020a). Similarity Analysis of Contextual Word Representation Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655, Online. Association for Computational Linguistics.

Wu, X., Dyer, E., and Neyshabur, B. (2020b). When Do Curricula Work? *arXiv:2012.03107 [cs, eess, stat]*. arXiv: 2012.03107.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019).

Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhang, K. and Bowman, S. (2018). Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium. Association for Computational Linguistics.

Zhang, Y. and Nie, A. (2020). Inducing Grammar from Long Short-Term Memory Networks by Shapley Decomposition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 299–305, Online. Association for Computational Linguistics.

Zhang, Y., Warstadt, A., Li, H.-S., and Bowman, S. R. (2020). When Do You Need Billions of Words of Pretraining Data? *arXiv:2011.04946 [cs]*. arXiv: 2011.04946.