UNDERSTANDING LANGUAGE MODELS THROUGH DISCOVERY AND BY DESIGN

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

John Hewitt
August 2024

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christopher Manning, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Percy Liang, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christopher Potts

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Diyi Yang

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

Abstract

Neural language models—probabilistic models of language defined by neural networks and often trained on very large datasets—are now widely deployed as the foundation of a range of natural language applications. Despite language models' widespread use, we do not have precise understanding of how they function. The behavior of these models is the result of trillions of words of text, billions of parameters, and GPU-centuries of compute. As a result, while we have recipes to make better models, we lack the precision of understanding to make surgical, targeted fixes to their failures. In this thesis, we present two lines of work towards deep understanding of language models. The first line is *understanding through discovery*: exploratory research in the *probing* paradigm that provides methods for discovery of what models know about properties of language. I show that language models learn and encode in their representations a considerable amount of knowledge about the syntactic structure of language. I also provide methodological improvements that help the field design and trust experiments like these. In particular, I show how probing can estimate *usable* information constructed by networks towards high-level properties. The second line of work is in *understanding by design*. While most work in understanding language models treats them as static, we as engineers have the ability to build them with the goal of understanding in mind, despite the challenge that their behaviors are learned from broad data. I present the Backpack, a neural network architecture that provides precise hooks for impacting model behavior. Finally, I show how to estimate the precision with which our methods for fixing problems in language models target desired behaviors without changing the model overall, in the model editing with canonical examples problem setting. All together, the methods and results in this thesis provide clarity of methodology and some progress in understanding language models, but also point towards how there is more left to be understood than that we understand.

Preface

What does it mean to understand a tool whose behavior is at least as much discovered as it is designed? Throughout my PhD, I've approached this question from two perspectives, often changing my mind about which I find more valuable, frustratedly vacillating as I tackled a problem as hard to define as to solve. Below I'll provide brief summaries of these perspectives to give you an idea of the background with which this thesis was written. However, despite the title of this thesis, **our understanding of language models is limited**. All I can offer, as is often the case in science, is a modest contribution and a lot of thoughts along the way.

The first perspective with which I tackled understanding is **blue sky exploration**—we observe properties of the tool in action, observe its internal components, observe its predictions, and make hypotheses about potential generalizations that could tune our intuitions about the nature of the tool. The core criticism of this perspective is, *to what end?* why bother; what does it get us? In engineering-centric fields, this is a natural question, especially since progress is measured and proven through concrete systems that perform better than existing alternatives. Other engineering fields certainly do fundamental research, and can rely on theory (e.g., from physics) more than we can in machine learning, but progress in machine learning has rarely been from, e.g., interpretability. In summary, why explore the blue sky if you can't explain how it'll help build something?

The second perspective on understanding I call **ability to repair as understanding**—we take on the guise of a technician, mechanic, or doctor, and leverage the strong intuition that real understanding implies the ability to localize a problem or suboptimality in a complex system to a set of malfunctioning components, and fix those components without negatively affecting the rest of the system. This intuition seems to make more natural sense than blue sky exploration to many in the engineering-focused NLP field, and is easier to evaluate. This begs the question of whether efforts in this direction are any different from non-understanding centric model improvement. In some sense, no! And to the extent that our "understanding" provides us no leverage to improve models beyond opaque methods, we should feel like our understanding is weak. Yet in another sense, repair as understanding isn't the same as model improvement, since it specifically measures our ability to make **targeted, or surgical** improvements: this is sometimes true in general model improvement, but is crucial in understanding work.

We are scientists and we are engineers, attempting to understand some of the most complex objects humanity has ever constructed, for the sake of fixing their faults, ensuring their safety, and developing more useful technologies. I hope this thesis proves useful in that goal.

Acknowledgments

Quite frequently, I am working, or walking, or travelling, and I am struck by how immensely lucky I am to have work that I find engaging and fulfilling, that supports my life, that is stable. I think at those moments back to my freshman year of college, in which I struggled immensely, cried often, and strongly considered dropping computer science. This would have been okay, but I am deeply grateful for where I have ended up instead.

I thank my high school teachers, including but not limited to Fran Guilbert, MJ Linane, Colin Everett, and Erich Carroll. Teaching high school is too often a thankless job in the United States, and in different ways each of these teachers helped me to push myself, which, as I found later in life, would continue to be a theme.

When I arrived at college, I was assigned Max Mintz as a course advisor, and I took an introductory discrete mathematics course with Rajiv Gandhi. These two professors pushed me and supported me even as I failed early on, and I don't think I would be in computing, let alone finishing a PhD, without them. Rajiv wrote me a letter of recommendation for a research program after I received a C+ in his discrete math course; I can only imagine it said "John tries very hard," and left everything else to pragmatics, since I got the position in the program. Ani Nenkova also wrote me a letter of recommendation for the same program, despite the fact that I had achieved very little as an informal researcher under her guidance; I'm grateful for her time and the fact that she took a chance on me. Likewise, I'm grateful to David Yarowsky for mentoring me that summer, and to Chris Callison-Burch, who then let me work in his lab for the rest of my undergraduate years. Eventually, my grades caught up to my peers', but I was never a stellar student, so I am beyond grateful that I was given opportunities to do research, which I ended up being more proficient at.

When I arrived at Stanford, I again felt like I wasn't going to cut it. I recall taking CS 229 (machine learning) my first quarter. I was surprised that I struggled with it (I was supposed to be doing a PhD in this stuff!) and I noted that a sophomore sitting next to me in lecture seemed to be picking up the concepts much faster than I. Again, it took time—years in some sense—to feel like I had "caught up" and still I am constantly impressed by how much more my peers tend to know than I. In that difficult first year, new friends I made outside my immediate academic surroundings—Steph, Micheal, Soumya, Lisa, Andrea, Travis, Rebecca—were a huge support and I thank them immensely. Andrea, you helped me learn how to have more fun. I'd eventually live with Travis and Rebecca for two and a half years, during the large part of the COVID-19 pandemic, and I shudder to think how I would have handled that time had it not been for their love and time. Travis is a source of constancy and sage advice, and I know no one who shows more love through planning events for her friends than Rebecca. With that whole group, years of thanksgivings, nights out, dinners, have been irreplaceable.

I thank the Stanford NLP community, a supportive and kind group of scientists whose insightful perspectives led to the refinement and improvement of much if not all of the work of this thesis. Thanks to my labmates, including but not limited to Rishi, Lisa, Stephen, Michael, Shiori, Robin, Jenny, Kevin, Ashwin, Siyan, Chris Donahue. Thanks to Shikhar, Moussa, Tolúlọpẹ́, and Dilara.

I had the good fortune of being "officially" in two separate lab groups, that of Chris Manning and that of Percy Liang, meaning I got to listen to a truly amazing range of talks and research projects. My cohort-mates—NLPers admitted the same year as me—Jesse Mu, Alex Tamkin, Isabel Papadimitriou (and Sidd, even though you joined a year later!)—I am glad we went through the tumult of the changing NLP world together. Within the group, I particularly appreciated the social efforts of Abi See, and tried to continue her legacy once she left. And I am thankful for the many takes and conversations with Nelson Liu over the years about life and research.

I'd like to thank Amita Kamath for being a first close friend in Percy's lab, sharing with me as we both struggled, and for the support in the years since. I wish you luck as you finish your own PhD!

Deepan, you've known me for a long time and seen me through considerable change; thanks for always being there, and for exploring a bit of the world with me.

In summer of 2022, I took an internship at DeepMind that helped me to recover from the pandemic and just enjoy travel, work, new places, new people. To my intern managers Adhi Kuncoro and Aida Nematzadeh, thank you for your support and letting me explore my odd ideas! To Yuqing, Ben, Lucio, Patrick, and Kayo, thank you for your willingness to go out, your willingness to talk and to laugh; what a special summer of friendship this was. Sanjana, I'm glad our coffee shop work—and discussion—schedule continued for years after London. Judy, thank you for your deep commitment to your friends and to building community, and for support in the final months of the thesis writing process.

Many of my friends over the years supported me in ways hard to describe here. Skyler, thank you for being a friend in San Francisco when I had few! Will Merrill, I'm sort of glad I couldn't convince you to come to Stanford given how much you love New York City, and I'm glad to have been able to chat with you when I've been there. Shreya Shankar, thanks for many movie nights and great conversations. Adam and Sami, thank you for Dungeons and Dragons nights. Jen, thank you for taking care of me when I was sick and for helping with my defense! Matt, we only lived together less than a year, but that year was packed with good times; thank you. Justin Smilan, it's tough to maintain a friendship across distances for more than a decade, and I'm glad we have. Thanks to Eric Zelikman, and to Cindy Wu, for being down for a vacation. Thanks to Clara Meister, for making friends from afar.

Thanks to Dan Jurafsky who in my first year, when I responded "not well!" to his question "how are you doing, John?" sat down with me and discussed his own struggles in the PhD and life, and made me feel welcome. Thanks to Surya Ganguli, for participating in a roller coaster research project, and for being my thesis committee chair. Thanks to my letter writers for academia: Chris Dyer, Chris Potts, and my advisors. Thanks to Tatsu Hashimoto; it's nice to have known you as a postdoc before you became a professor, and to now join you in that pursuit.

Thanks to the Stanford CS 224n TAs that I worked with, especially those when I was head TA. That course is a huge undertaking, and would not be possible without the efforts of these TAs.

Thank you to my non-advisors on my reading committee, Diyi Yang and Christopher Potts. You both have been amazing supports and sources of discussion and refinement for my research. Thank you for opening your calendars and for helping me get across the finish line; you went above and beyond!

To my advisors, Chris Manning and Percy Liang, I feel like I've learned an immense amount under your guidance. You gave me freedom to do the research I wanted, you gave me honest opinions about what you felt would work and what you felt would be impactful. You gave me space and time to be unproductive when I needed it. As I think about becoming a professor myself, I have been thinking about how much of a commitment it is to take on a PhD student. You ended up talking to me more or less once a week for six years, and I am so honored and thankful that you took that time on me. Percy, I credit to you the confidence I take into writing papers with more rigor, more precision. Chris, I credit to you the ability to take a research idea and ask who will find it useful, why, and how we can improve on that. I've always been a bit of a scatter brain, especially when it comes to administrative requirements and deadlines; thank you both for your patience. I hope to pass on the knowledge you've helped me develop to generations of students to come.

To my family, thank you for giving me love and support and freedom. My siblings, Sarah, Steve, and Mike, my dad Andrew, and my stepmother, Susan—you all have given me a lot of space and time—as well as support—to do my own thing, and for that I am grateful! To my mom, Sandra, who died when I was nine, you taught me how to read, and led me to read voraciously. You taught me to be deeply thankful at all times, as you were, even though the situation you were in would deplete the thankfulness from almost anyone. Thank you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A neural language model p_θ is a probability distribution over strings—or parts of strings—from a finite vocabulary $\mathcal{V}$ that is defined by a neural network with learnable parameters θ. In some sense, it is much more than that. Empirically, the parameters θ learned for language modeling are a great starting point for finetuning—making small updates—to a model towards a range of tasks. The internal representations $\phi_\theta(\boldsymbol{x})$, constructed by neural network ϕ with parameters θ, and where $\boldsymbol{x} \in \mathcal{V}^*$, encode exceptionally high-level, general, useful properties of texts. And increasingly, models are given instructions explicitly with little updating, and perform competently across a range of tasks. Neural language models and related systems form the foundation for almost all modern natural language processing practice, and they empirically learn a radical amount about language and world. This knowledge is piecemeal, unsystematic, fuzzy, overwhelmingly large, adaptable, biased, unpredictable, and to me, fascinating. Language models are a dual artificial-natural phenomenon that are worth deeply understanding. Understanding means many things, so it's useful to begin with an example.

1.1 On what we expect of our technologies

In the 1990s, Thomas Nicely discovered that certain Intel Pentium processors were returning the wrong answers to certain floating point division operations. That is, sometimes you ask the CPU for x/y and instead of returning the correct floating point approximation of that, it would return another value (with varying amounts of error.) This was a big deal. We expect precision out of our processors; they have specifications that we expect them to meet, and getting something as simple as a floating point operation wrong is a huge problem. (Though, this particular error would only occur for some inputs, which is seemingly part of why it took a considerable amount of searching and confirming to determine it was, in fact, an error of the processor.)

This bug damaged Intel's reputation. However, processors are *technologies*, and we understood these Pentium processors well enough to localize the source of the error to a programmable logic array (PLA) that wasn't written into correctly. Whenever that specific PLA was accessed in order to compute that particular floating point division, the wrong values would be accessed and the returned value would be wrong. This localization led to a very clear fix: if you fix the PLA, you fix the division. Future generations of Intel's processors did *not* have this bug.

So, we expect technologies to work as specified. But when they don't, we expect our technologists to understand the technologies well enough to develop precise, targeted fixes to those problems. By fixing the PLA, we didn't, say, make the rest of the processor slower, or potentially cause another bug somewhere else. All of this was enabled by our understanding of how the CPU works.

Now compare this process to what we expect of language models. In 2022, New York Times journalist Kevin Roose interacted with Microsoft's Sydney chatbot[1]. After a long interaction, the chatbot went a bit off the rails, professing its "love" for the journalist and encouraging them to leave their spouse. This is a bug in the chatbot, which to my understanding was already an extensively tested, expensive beta product.

The story caused considerable negative press for Microsoft. Whereas Intel could localize its bug to a PLA in its microprocessor, what could Microsoft do to localize this bug to a component of its neural language model chatbot? Unfortunately, our tools for understanding and fixing language models are coarse, and no localization was really possible. Certainly, Microsoft could re-do safety tuning, changing all or many of the learnable parameters of the model to try to do this sort of behavior less, but this could have effects on the rest of the model, and/or not actually fix the bug. To be fair, this is an exceptionally hard problem: language model behaviors are complex and derive from trillions of words of text and GPU-centuries of computation, not directly from our specifications. But still, to be considered technologies, we need our tools to be fixable precisely. This derives from an understanding of the model at some level, and we're just not there yet.

In the thesis ahead, I present five separate works, each of which having this goal of understanding in mind. Later in this section, I will give a brief overview of each research work. I will not attempt to plainly summarize; most of the works are at least a few years old, and so as much as I am able I will recontextualize them in a world and research landscape (in 2024) that already seems very different from those in which they were written (e.g., in 2018.) To do so, it is worth it to begin with some context.

1.2 A bit of background for context

1.2.1 Non-neural language models

When I began working on natural language processing—the science and engineering of systems that operate with and/or generate human languages—in 2015, I was introduced to language models as a tool with a very specific purpose in my subfield: when multiple translations of a phrase (i.e., from English to Tamil) are possible, a language model would score the phrases as to how *likely they were to be good Tamil*, independent of whether they were good translations of the English phrase. In a world where single-language data was (and is) much more prevalant than pairs of sentences in two languages with the same intended meaning, this meant that the language model could benefit from more data than the translation model. The language models used

[1]`https://www.nytimes.com/2023/02/16/technology/bing-chatbot-microsoft-chatgpt.html`.

were *big*; they stored every single sequence of three to five words in a large text corpus, like *I like pizza*, or *like pizza because*, or *pizza because it*, and stored the counts more or less exactly. They also were quite transparent in what they did. Estimated probabilities were essentially derived from literal counting:

$$p(\text{eat the pizza} \mid \text{eat the}) \approx \frac{\text{count}(\text{eat the pizza})}{\text{count}(\text{eat the})}, \tag{1.1}$$

though in practice these models were *smoothed* (a process in which a small amount of probability mass is placed on unobserved elements.) These simple systems also afforded the generation of text by sampling from the estimated distributions. Despite the simplicity of these count-based models, the generated text could be quite coherent in short spans (Goldberg, 2017; Jurafsky and Martin, 2000; Hewitt et al., 2022) – especially if models were not smoothed.

These systems were undeniably useful—improving language models was a core part of improved automatic translation between human languages as well as automatic speech recognition of audio at the time—but did not form the substrate upon which language understanding technologies were built at the time. They scored text relatively well, and that was it. And if we wanted to *understand* these count-based models in a meaningful way, we could quickly come to a conclusion. A given score could be traced to the exact strings that were counted (and perhaps the smoothing that was used.) In summary, these count-based models[2] were not artefacts whose behaviors were powerful and opaque enough to spur the attempts at understanding we've participated in in this thesis.

This begs the question of what changed. At first glance, it wasn't the introduction of new ideas for how to build language models.[3] Neural language models already existed (and had since at least the work of Bengio et al. (2000), though arguably also since the work of Elman (1990).) In fact, the technologies we use today are rather similar to those of Elman (1990). The core difference is the massive scale at which neural language models have been constructed in the last decade—and the surprising difference in qualitative outcomes that happen when you scale neural language models compared to scaling count-based models. I believe this distinction to be underappreciated: before the modern language model revolution (say, 2018 onward,) NLP practitioners had already spent huge efforts scaling count-based language models, and the results were *fine*. Nothing unpredictable happened; the models became huge and machine translation or transcription performances became a bit better. But what has happened as we've scaled neural language models, with distributed representations (which we'll discuss soon), is just massively different.

1.2.2 Neural Language Models

There is no perfect definition for a neural network that includes all things that feel like neural networks and excludes all things that feel like not neural networks, and so there is no perfect definition for a neural language

[2]These count-based models are often called n-gram models. We avoid that language here since it refers to a finite window of n words of memory, not to the method of estimation, that is, counting.

[3]To be sure, a huge amount of excellent work has gone into refining the learning techniques and the computer software and hardware systems, which we'll get into later.

model. For our purposes, thankfully, intuition will suffice. A neural language model, intuitively, is a language model p_θ wherein (1) the parameters θ are learned via some variant of gradient descent, (2) where there is some kind of nonlinear function applied to an input in order to come to an output probability, and (3) whereby inputs are represented as real-valued vectors. Here's a nice example using a recurrent neural network:

$$h_0 = \mathbf{0} \tag{1.2}$$

$$h_t = \sigma(W_h h_{t-1} + W_x x_t) \tag{1.3}$$

$$p_\theta(\cdot \mid \boldsymbol{x}_{<t}) = \text{softmax}(E h_{t-1}) \tag{1.4}$$

where all $h_i \in \mathbb{R}^d$, and $W_h \in \mathbb{R}^{d \times d}$, $W_x \in \mathbb{R}^{d \times |\mathcal{V}|}$, $E \in \mathbb{R}^{|\mathcal{V}| \times d}$, and $x_t \in \mathbb{R}^{|\mathcal{V}|}$ is a one-hot encoding of a vocabulary element of $\mathcal{V}$. The parameters of this model are $\theta = \{W_h, W_x, E\}$. They could be trained by gradient descent to minimize the cross-entropy prediction error on samples drawn from some distribution $\mathcal{D}$ over $\mathcal{V}^*$, which we'll write as tuples of tokens x_t and the prefixes preceding them $\boldsymbol{x}_{<t}$:

$$\min_\theta \mathbb{E}_{(x_t, x_{<t}) \sim \mathcal{D}} \left[-\log p_\theta(x_t \mid \boldsymbol{x}_{<t}) \right] \tag{1.5}$$

When discussing scaling neural language models, two things are meant, often jointly: scaling the number of learnable parameters (i.e., $|\theta|$, if we were to flatten and concatenate all matrices in θ), and scaling the number of samples trained on (i.e., the number of tokens x_t and corresponding prefixes $\boldsymbol{x}_{<t}$ that participate in the expectation in Eqn 1.5.) More learnable parameters correspond roughly to increased capacity (and, perhaps surprisingly, easier learning) in the neural network, and more samples both provide more knowledge and seemingly force models to learn more interesting functions from input sequences to output distributions. Note that for count-based models, we can also scale parameter counts (i.e., by counting ever-larger spans of text) and samples (by counting over a larger corpus.) Intuitively, though, the algorithms used by these count-based models to estimate probabilities—i.e., counting and division—are similar no matter the scale.

In contrast, neural language models seem to learn very different functions at large scale than at small scale. The complexity and depth of this result will take much more than the length of this thesis to go into. So, I'll provide an example of this scale at something like its simplest, with a model that (1) is often (erroneously) not considered a language model, and (2) is often (reasonably) not considered a neural network: continuous bag-of-words word2vec (Mikolov et al., 2013b,a).

1.2.3 Word2vec as a neural language model

Continuous bag-of-words word2vec was introduced as an efficient method for learning useful representations of single words (i.e., single elements of our vocabulary $\mathcal{V}$) (Mikolov et al., 2013a). A not-so-useful representation is the one-hot vector, $v_x \in \{0,1\}^{|\mathcal{V}|}$, in which each unique word has its own dimension where the value is 1, and 0 elsewhere. This representation is not-so-useful because two different words that feel similar—say, *house* and *home*—are equally as dissimilar as any other pair of words under a similarity function like the dot product.

Relationship	**Example 1**	**Example 2**	**Example 3**
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein: scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Table 1.1: Table 8 of Mikolov et al. (2013a). The original caption was *Table 8: Examples of the word pair relationships, using the best word vectors from Table 4 (Skipgram model trained on 783M words with 300 dimensionality).*

In contrast, continuous bag-of-words word2vec vectors encode interesting similarities between words via the dot product.

I'll now present continuous bag-of-words word2vec, (which we'll see again in a modern model in Chapter 5.) For technical reasons, I'll use a slightly different mathematical form than presented in any of Mikolov et al. (2013b,a). For a sequence of words $(o_1, \ldots, o_{k//2}, x_i, o_{k//2+1}, \ldots, o_k) \in \mathcal{V}^*$, continuous bag-of-words word2vec specifies a distribution over the inside word x_i given the outside words $o_1, \ldots, o_k$:

$$v_i = Ex_i \tag{1.6}$$

$$v_o = \frac{1}{k} \sum_{o_1,\ldots,o_k} Eo_k \tag{1.7}$$

$$p(x_i \mid o_1, \ldots, o_k) = \frac{\exp(v_o^\top v_i)}{\sum_{x\in\mathcal{V}} \exp(v_o^\top (Ex))} \tag{1.8}$$

You may note that this looks a lot like a simplification of our recurrent neural network language model in Equation 1.5. Fundamentally, it is quite similar; in fact, by predicting the last word instead of the center word, we see that this is a variation on a simple language model. Let $(x_1, \ldots, x_t) \in \mathcal{V}^*$. Then

$$v_t = Ex_t \tag{1.9}$$

$$v_{<t} = \frac{1}{t-1} \sum_{i=1}^{t-1} Ex_i \tag{1.10}$$

$$p_\theta(x_t \mid x_{<t}) = \frac{\exp(v_o^\top v_i)}{\sum_{x\in\mathcal{V}} \exp(v_o^\top (Ex))}, \tag{1.11}$$

and we optimize the parameters $E \in \mathbb{R}^{d\times|\mathcal{V}|}$ to minimize an identical loss as Equation 1.5. Is this a neural language model? It depends on your definition of "neural network", but I'm happy to go with *yes* for now.

I am happy to call word2vec a neural language model in part because it is the simplest model I've seen that has some of the fascinating results of scaling neural language models. What properties should we expect out of word2vec as I've stated it? For one, we should expect that when two words are likely to co-occur, they should have similar vectors under the dot product (relative to the similarities of other words' vectors)[4]:

$$x, x' \text{ co-occur} \Rightarrow (Ex)^\top Ex' \text{ is large} \tag{1.12}$$

However, at scale, something somewhat surprising happens. Various relationships are approximately represented by vector offsets in word2vec embedding space:[5]

$$v_{\text{Einstein}} - v_{\text{scientist}} \approx v_{\text{Messi}} - v_{\text{midfielder}} \tag{1.13}$$

$$v_{\text{Einstein}} \approx v_{\text{Messi}} - v_{\text{midfielder}} + v_{\text{scientist}}. \tag{1.14}$$

In Table 1.1, I present what was Table 8 of Mikolov et al. (2013a), in which a linear offset (and exclude input words) strategy in word2vec vector space yields fascinating analogy completions like the Einstein-Messi relation above. Co-occurring words leading to similar vectors (Equation 1.12) is predictable; I would argue that approximate linear analogies (Equation 1.14) is a much harder property to predict would happen when shown the optimization problem being approximated by word2vec. Certainly, a lot of work has gone into understanding *why* this and other interesting properties arise in word2vec (Ethayarajh et al., 2019; Pennington et al., 2014; Levy and Goldberg, 2014).

The radical capabilities and problems in language models—the motivation for deep understanding—result from an almost identical optimization problem as I've presented here for word2vec; the only clear distinction is in the scaling of the representational capacity of the neural network and in the amount of data and computation put into the minimization problem.

1.2.4 Broad overview of related work

Each of the chapters of this thesis has a related work section specific to the topics of the chapter, but it bears briefly discussing the broader machine learning literature here at the beginning.

Interpretability. Multiple large bodies of work in machine learning have targeted the goals of *explaining* or *interpreting* the predictions of machine learning models. Interpretations and explanations can take many forms. Influence functions try to answer the question of which training data points were most relevant to classifying an example (Koh and Liang, 2017). Many works attempt to derive explanations in the form of which input words (or pixels, etc. in other modalities) are most relevant to the prediction of a machine learning model, also

[4]In the original word2vec formulation, in which different vectors are used for conditioning vs predicted words, this claim doesn't quite hold; instead, one expects words that co-occur with similar words to have similar vectors.

[5]This claim has garnered some controversy within NLP. One must exclude the words in the input—i.e., *messi*, *scientist*, *midfielder*—for the nearest neighbor to be *Einstein* instead of one of the input words (Linzen, 2016). Functions other than linear offsets extract analogies better as well (Rogers et al., 2017). Despite all of this, however, when one simply plays around with word2vec vectors and discovers, e.g., that the "linear offset plus exclude input words" method maps *France:wine::Australia* to *beer*, or listens to students as they play with the vectors and interesting analogies that are encoded, I just can't help but feel like the properties that *are* approximately encoded as linear-ish offsets are too fascinating to ignore.

called saliency maps (Simonyan et al., 2014; Ribeiro et al., 2016; Li et al., 2016). The difficulty of defining interpretable or explainable systems, and the difficulty of evaluating interpretations when we lack ground truth, has led both to influential pieces on interpretability as a goal (Doshi-Velez and Kim, 2017; Lipton, 2018), and concrete methodological issues with seemingly intuitive methods (Bilodeau et al., 2024; Geirhos et al., 2024). Much of the work in this thesis will attempt to leverage existing concepts in, e.g., lingusitics, in order to better understand representations in neural networks; this is common more broadly in machine learning (Kim et al., 2018; Koh et al., 2020). Even more broadly, using concepts or symbols as a substrate for understanding has a history in neurosymbolic computing; in my thinking on the subject, I found the key-value binding ideas in tensor product representations (Smolensky, 1990) particularly influential.

Connections to linguistics in interpretability. This thesis leverages, at least indirectly, decades of work in linguistic and computational linguistic study on the structure and processing of language in humans. Core to the first part of this thesis is the idea that human language has latent, syntactic, hierarchical structure. I leverage linguist-parsed resources like the Penn Treebank (Marcus et al., 1993) and syntactic formalisms like Universal Dependencies (Nivre et al., 2020). These resources are compromises between linguistic study and ease of annotation and computational use. Ideas surrounding the characterization of the structure of language are much older (Humboldt, 1836; Chomsky, 1957; Montague, 1970), and the exact formalisms are not crucial for this thesis. Separately, ideas often traced to Firth (1935, 1957) about the meanings of words being derived from the distribution of contexts in which they appear are central to the ideas in language modeling.

Neural Architectures. In part 2 of this thesis, I introduce the Backpack, a neural network architecture. The development or application of neural network architectures for processing language, with its variable lengths and structures, has a long history, including e.g., recurrent neural networks (Elman, 1990) and gated variants like LSTMs and GRUs (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). Attention for long-distance dependencies and gradient propagation has been one of the most influential ideas in the last decade Bahdanau et al. (2014); Vaswani et al. (2017). Recently, state-space models, often with gating, have risen in popularity due to their linear-time complexity in sequence length Gu et al. (2021); Gu and Dao (2023). Ideas like the mixture-of-experts have risen and fallen in popularity, a sort of meta-architectural decision that is now common in modern language models Shazeer et al. (2017). Most architectural development is for the sake of more performant models; the Backpack is designed for our understanding, but can leverage other improvements, since it can be thought of as a constraint on top of other neural architectures.

1.3 On understanding

Understanding is in the mind of the understander. I discussed this somewhat in the preface to this thesis, but now that we've discussed a bit of detail of a model—word2vec—that we could attempt to understand, it is useful to revisit.

As researchers, it is our job to take complex, hard-to-define problems, and bring to bear on them all of our intuitions, our tools, our hypotheses. There is no one tool that solves these problems out of the box, no one intuition that applies perfectly. Often research conversations go along the lines of *what would you think would happen if X?* and the response *well I'm not sure how relevant it is, but I know that Y, and the other day I realized that* Z.... The future value of each piece of knowledge is exceptionally hard to quantify, as one doesn't know when it will be relevant or what other piece of knowledge is necessary to make the piece of knowledge actionable. Yet, we strive after this understanding as researchers in order to prune the exponentially large search space of actions as a function of time we could take as we attempt to discover truths about the world and build systems that are useful.

As engineers or technicians, it is our job to have a causal understanding of what makes our systems tick, to fix their problems, specify their boundaries, characterize their known issues. Knowing which part of a system to fix is a basic requirement of any technician. When an airplane crashes, or when a microprocessor has a bug, we expect the engineers that designed the systems to come up with clear, exact changes that will make such failures massively less likely in the future.

Understanding is required for all of this. It is easier to value the engineer's, the technician's understanding. It is immediate. Many results in our field, like better recipes for training models, or methods for improving benchmark numbers, fall under this category. However, I believe the vague intuitions and understanding of the researcher are crucial for long-term progress. Each researcher will decide which intuition, which result helps them think more clearly about the artefacts they're building and studying, which is why I say understanding is in the mind of the understander.

This more vague, blue-sky understanding, is as much about the journey as it is about the result. Imagine if, in a zero-knowledge proof, an oracle were to convince you that P$\neq$NP, that is, that the set of problems that are efficiently checkable is strictly larger than the set of problems that are efficiently solvable. Depending on the amount of uncertainty you had in that statement, you've received less than (perhaps substantially less than) one bit of information about the world. I imagine most computer scientists already mostly believe this, but this oracle removes all doubt, and tells you nothing about why the claim is true.

Compare your feeling in this hypothetical to if you were to receive the proof and it contained a beautiful *reason* why P$\neq$NP. You've gained no more information about the truth of the claim than in the zero-knowledge setting, but you've gained potentially useful information about so many other things. I imagine that this would change your thinking about future problems, and you could bring the intuitions you've developed to bear on even only tangentially related ideas. This is the beauty of conducting blue sky understanding research; it builds abstract tools in the toolboxes of researchers who may apply them scenarios you may not even dream of.

1.4 Understanding language models through discovery and by design

With all of the context of this chapter so far in mind, I can explain why the thesis of this work is to understand language models through discovery and by design. To refine this even further, what I attempted to do in my thesis work is to develop and refine concrete methods evaluation frameworks for formalizing and measuring aspects of what it means for a language model to have knowledge of a concept, or for us to be able to make surgical changes to a model. I provide concrete results—like how neural language models learn about natural language syntax without being explicitly taught—but my main goal has been to think through the problems relating to what it means to understand these exceptionally complex systems and develop methods that allow us to tackle those problems. Implicit in my thesis is that both of my proposed directions—understanding through discovery, and separately, understanding by design—are necessary for long-term progress in our understanding of systems. On the one hand, this might seem uncontroversial; we don't stop doing basic biology research just because we're also doing applied drug design; we need to keep doing all levels of research to ensure the pipeline of scientific results for centuries to come. But on the other hand, in the immediate results-oriented engineering fields in which machine learning methods have flourished, it's not an obvious thing. Understanding through discovery is necessary because we just don't know the right way to characterize neural language models or how they store their knowledge; we need to just explore and make connections. Understanding by design is necessary because these systems are already broadly deployed, and as engineers we cannot build systems and hope to understand them later; we must build them with our ability to understand and fix them in mind. I can't prove the necessity of both directions in this thesis, but I do hope that the reader learns qualitatively different things from the two core halves of this thesis and thus comes away somewhat convinced. With this in mind, I'll now discuss a bit about each of the attempts at understanding that you'll find in the rest of this thesis.

1.5 Chapters in this thesis

1.5.1 A Structural Probe for Finding Syntax in Word Representations

In Chapter 2, which corresponds to the published work Hewitt and Manning (2019), we showed that neural language models—in particular, BERT (Devlin et al., 2019)—learn to encode a fascinatingly simple representation of linguistic syntax, simply by an optimization like that in Equation 1.5: predicting words in a sequence. Most of the interest of this work was not in the result itself, but in the methodological development. How does one show that an artefact has learned a property other than what it was trained on? How do we connect the continuous, vector representations of language models with the discrete, graph structures hypothesized by linguists?

First, a note on syntax. Natural language has long-distance, hierarchical structure. Consider the following sentence:

The chef who ran to the store was out of food.

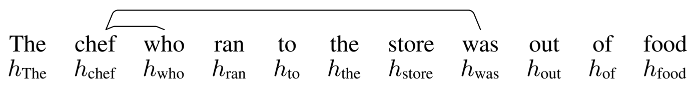
There's a substring here that reads as a whole sentence itself, and leads to the wrong interpretation of the sentence, in which the store is out of food instead of the chef:

The chef who ran to the store was out of food.

Instead, the correct semantics (the chef being the one out of food) is derived by realizing that *who ran to the store* modifies *chef*, and the chef was out of food:

> *(The chef (who ran to the store)) (was out of food).*

These relationships can be represented as a graph, where the words are nodes and edges represent modification relationships, a few of which I've written here:


The chef who ran to the store was out of food
h_{The} h_{chef} h_{who} h_{ran} h_{to} h_{the} h_{store} h_{was} h_{out} h_{of} h_{food}

In contrast to this discrete structure, a model like BERT represents words (technically, parts of words) as real-valued vectors $h_{\text{chef}} \in \mathbb{R}^d$ etc. The key insight of this work is that there is a natural connection between the discrete tree structure of syntax and the continuous space of neural activations: they both correspond to natural metrics. The path metric of a tree is the number of edges in the unique path between each pair of nodes. Various natural distances exist on embeddings, like L2 or L1. In fact, there is a beautiful equivalence between L1 metrics and path metrics (Deza and Laurent, 2009). In this sense, these objects are not so different.

The final component of the method we use is learning—we learn a linear transformation of the BERT vector space under which the distances in linguistic trees and the distances in vector space are approximately the same. The learning uses real syntax trees to find a good such linear transformation, and it should cause some discomfort scientifically—if we're *learning* the mapping, how do we know we can't just find, well, anything? We tackle this question in multiple of the following chapters.

1.5.2 Designing and Interpreting Probes with Control Tasks

The methods in Chapter 2 fall under a banner called *probing*: learning a mapping from the representations of a model to a property of interest in order to understand those representations. Upon finishing the paper corresponding to that chapter, I thought more deeply about why I thought the results were interesting. Was it interesting just that there existed *some* mapping from BERT vectors to syntax trees? What was the difference, if any, between the scientific effort of probing to understand the representations and the engineering effort of figuring out how best to use a representation to predict a property by any means necessary?

At a high level, this is not a problem new to science. Neuroscientists in particular have dealt with questions of what it means for a part of the brain to encode a particular property, often assuming linear mappings as proof of an interesting level of encoding, though this has received increasing attention of late (Ivanova et al., 2022). As put to me by Stanford theoretical neuroscientist and deep learning theorist Surya Ganguli, the neurons in the eye obviously encode the information of, say, a lion, but it's represented at a very low level, whereas deeper in the brain, it's represented at a higher level. I'm paraphrasing this heavily, and I think it is a common enough analogy; it was also used by, e.g., Alain and Bengio (2016).

Simplicity of the decoding mapping seems key to me – though simplicity can be defined many ways.

Around 2019, however, NLP scientists running probing studies were using rather expressive mappings—multiple-layer neural networks—and I felt like in order to make a point, I needed clear evidence that such methods could lead the researcher astray. This chapter, corresponding to the published paper Hewitt and Liang (2019), provided that evidence. At its core is a fake task—basically memorizing which randomly chosen categories each word in a vocabulary belongs to—which (1) the representation we're trying to understand definitely has no special knowledge of, and yet (2) through probing experiments, we can be reasonably convinced via a held-out test set that the representation *does* have that knowledge. All of the "hard work" in some sense was done by the probe, demonstrating the pitfalls of attributing interesting properties to the representation.

The saving grace for probing, to me, is that simple probes really don't have this problem in comparison to complex probes. We introduced a sort of baseline—controlling for this random task accuracy when probing for something else—which in retrospect is a bit of a kludge; we sorted out the right thing to do only in the subsequent chapter (below.) But the key insight to me was that the simplicity of the mapping one attempts to find was crucial in avoiding the kind of spurious result that more complex probing mappings could allow.

1.5.3 Conditional Probing: Measuring Usable Information Beyond a Baseline

Chapter 3 provided a clear example of how an expressive probe could lead a researcher to an incorrect conclusion about the properties of a representation they're attempting to study. However, it didn't provide a theoretical framework or set of methods for probing that would allow us to avoid, or design away, these problems.

In this chapter, corresponding to the published work Hewitt et al. (2021), we provided an argument and theoretical framework for the idea that probing measures *usable information* (Xu et al., 2020) under a researcher's hypothesis for what *usability* might mean – linearity, some kind of kernel function, etc.[6] To understand usable information, consider an encrypted document that contains an address. The encrypted document contains all the information of the address, and if you worked hard enough—say, took exponential time—to decrypt it, you could get the address! But the information isn't very *usable*. Usable information quantifies this by specifying a class of functions by which one could attempt to use, say, the document to predict the address. Often this class might be *linear models*, but for now, think of it as a class of functions representing a human trying to read the encrypted document. The usable information is the most uncertainty one could remove from what we think the address is, under the best function in the class we've chosen. Naturally when a human tries to read an encrypted document, we're just as uncertain after seeing it as we were before. For an unencrypted document, the information is quite accessible to us reading it.

In this chapter, we introduced conditional usable information. Much like conditional Shannon information, conditional usable information measures how much extra uncertainty is removed by knowing something (say, the unencrypted document) conditioned on already knowing something else (say, the encrypted document.)

[6] I designed, led, and implemented most of the work in Hewitt et al. (2021), with useful contributions in proofs and discussion from others.

Neural language models, in this analogy, are the decryption function; they take information that is present in the input, and make it *usable* to a wide range of simple extraction functions.

Making a property easier to predict to weaker classes of functions is, to me, a deep notion of having learned about that property. There's certainly room for disagreement on this, but the more I ponder what it means for an artefact to contain knowledge, the more it seems right. An agent that has knowledge can try to convince you of that fact. An artefact that has knowledge is not trying to convince you, so you need to decide how hard you want to try to read it. A library contains knowledge; an ancient tome contains knowledge. Neither tells you how to get that knowledge or attempts to convince you of that knowledge. Likewise a neural network contains knowledge.

This line of three Chapters—2,3,4—corresponding to the papers Hewitt and Manning (2019), Hewitt and Liang (2019), Hewitt et al. (2021)—are Part 1 of this thesis. They are all are attempts to understand neural objects that weren't designed with understanding in mind. But we are engineers, not just scientists, and we can design future neural systems. The next two chapters explore the idea that neural language models can be designed for the sake of understanding.

1.5.4 Backpack Language Models

In Chapter 5, corresponding to the published work (Hewitt et al., 2023), we present the Backpack, a neural network architecture designed for our ability to understand it as well as its power for learning from data.[7] Existing language models don't really have gauges or control panels like traditional tools. The biggest reason is that the things we might like to measure or intervene upon can't really be specified by the designers a priori. High-level concepts that we might want to intervene upon will be learned by the system from the data; any attempt to specify them a priori is likely to (1) degrade system performance since we don't allow the data to speak as clearly, and (2) perhaps not give us the control we want, since the model might learn to use the concept we've tried to specify in a way we didn't predict.

The approach we take in the Backpack is to specify a specific functional form of a deep neural network that (1) builds in knob-like components, but (2) forces the network to learn the semantics of those knobs from scratch from the data. The particular math we use enforces that the knobs are knobs: they can be turned up or down with reliable effects. This is rare in neural networks, wherein changing any parameter or activation could in theory have more or less *any* effect on new inputs.

More concretely, the Backpack constrains the deep network's output to be a non-negative combination of a set of word-identified factors. In a way we make precise in the chapter, it's like word2vec specified k vectors per word, and a controller network got to decide which factors of which words in the input were used to predict the next word. These lexical factors specialize in practice to decompose the predictive uses of different words, and these specializations don't depend on anything but the identity of the word. So, they provide the knobs that we can twist, and we get a guarantee of knob-like behavior because the only thing that changes in a factor

[7]With useful discussions and writing from my coauthors, I designed most of this work and implemented all code and experiments (leveraging existing codebases as noted.) The exact form of the Backpack in particular was jointly designed with John Thickstun.

is how much it's used in a particular context, not what its use means for the output distribution.

The experiments in this chapter demonstrate that the Backpack is expressive like the Transformer (Vaswani et al., 2017) is, and that the sense vectors provide some interesting new ways to control the model's behavior. It was in this chapter that I began to think about interpretability-as-control: beyond blue sky exploration of a model, we can evaluate our understanding of our models by our ability to (1) change what we want about the model while (2) leaving everything else unchanged. This is a deep measurement problem: there's so much that could change that we don't want to change, and there's so much that we could measure to see if we've changed everything that we did want to change. Our experiments in this chapter begin to touch on this, but I felt the problem was deep enough that I spent Chapter 6 investigating it.

1.5.5 Model Editing with Canonical Examples

I am no medical doctor, but I have the intuition that there are two goals to a good medicine: (1) fix some problem with the body, and (2) don't break anything else. I'm reminded of Randall Monroe's xkcd comic, number 1217: *Cells*, which contains the text:

> When you see a claim that a common drug or vitamin "kills cancer cells in a petri dish," keep in mind: so does a handgun.

This quote gets at the core of how *surgical* (in the colloquial, but also maybe technical, sense) we would like interventions to complex systems to be when we're fixing a problem. Often, we do basic science to understand biology or the human body better in order to eventually use that understanding to do a better job of making medicines that follow both goals (1) and (2). Science that doesn't immediately lead to better medicines can of course still be valuable; I've already argued as much and I don't intend to downplay that. But when we are *attempting* to develop better medicines, we must carefully measure both of our constraints.

In Chapter 6, corresponding to the preprint work Hewitt et al. (2024), in which I designed, led, and implemented most of the work with help in implementation of some of the code and datasets, we attempted to develop a set of rules—an evaluation setting—under which we could measure how surgical changes to language models are across a wide range of possible things we'd like to change about them. Our setting has a few things I think are crucial. The setting gives a dataset of strings—good strings, bad strings, pairs of better and worse strings—that specify some simple version of what we'd like to change. A statement of fact, or a bias we'd like to remove from the model. The practitioner's job is to take a model and those strings, and return a new model. The first thing we require is that the returned model is with a tiny ϵ-ball of the original model in terms of overall loss on a large corpus; this means *on average*, very little can have changed in the model. The second thing we require is that the evaluation examples for the target behavior (i.e., knowledge of a fact, removal of a bias) be *out-of-distribution*, and in particular, harder, than the training examples, since all real evaluation in naturalistic contexts will be out-of-distribution.

Within this game, we find perhaps surprisingly that simple methods—LoRA finetuning with KL-divergence regularization (Hu et al., 2022)—perform quite well compared to methods that were designed specifically for their surgical purposes. This challenges how much our insight has really gained us in terms of our ability to surgically change the model compared to methods that were not developed for interpretability. (Though we do find that the methods enabled by the Backpack are even more useful.) At a broader level, this focus on whether our interpretability methods help us to make more surgical changes forces us to grapple with what our interpretability methods give us. In some sense, saying that finetuning actually can be more surgical than our editing methods is like getting our zero-knowledge proof of P$\neq$NP as I discussed earlier in the chapter; we know that this thing works, but we still don't really know why. It's not satisfying, but it is effective.

1.6 On the rest of the thesis

The rest of this thesis is best read as Chapters 2 through 4 together (Part 1), and/or Chapters 5 and 6 (Part 2). The first three chapters go into the deep methodological questions of probing, and some of the exciting results surrounding what language models learn about the structure of language without any supervision. From the title of this thesis, these chapters are on *understanding through discovery*. Chapters 5 and 6 document our efforts to make and measure language models that are understandable and controlable by design. Again from the title, these chapters are on *understanding by design*.

One might complain that these are really two quite distinct lines of research, and alas, they are. But I hope through this introduction that I've convinced you that they follow from two pillars of understanding—blue sky exploration and engineering-level control—both of which are fundamental and complementary. As I note in the preface, our understanding of language models is limited, despite my best efforts, and I hope the attempts I've documented here are useful for you as you continue on your own journeys of understanding.

Part I

Understanding through Discovery

Chapter 2

A Structural Probe for Finding Syntax in Word Representations

2.1 Introduction

As pretrained deep models that build contextualized representations of language have been shown to provide gains on NLP benchmarks, understanding what they learn is increasingly important. To this end, probing methods have been designed to evaluate the extent to which representations of language encode particular knowledge of interest, like part-of-speech (Belinkov et al., 2017), morphology (Peters et al., 2018a), or sentence length (Adi et al., 2017). Such methods work by specifying a *probe* (Conneau et al., 2018; Hupkes et al., 2018), a supervised model for finding information in a representation.

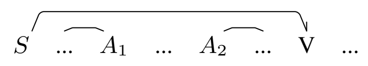
Of particular interest, both for linguistics and for building better models, is whether deep models' representations encode syntax (Linzen, 2018). Despite recent work (Kuncoro et al., 2018; Peters et al., 2018b; Tenney et al., 2019), open questions remain as to whether deep contextual models encode entire parse trees in their word representations.

In this chapter, we propose a *structural probe*, a simple model which tests whether syntax trees are consistently embedded in a linear transformation of a neural network's word representation space. Tree structure is embedded if the transformed space has the property that squared L2 distance between two words' vectors corresponds to the number of edges between the words in the parse tree. To reconstruct edge directions, we hypothesize a linear transformation under which the squared L2 norm corresponds to the depth of the word in the parse tree. Our probe uses supervision to find the transformations under which these properties are best approximated for each model. If such transformations exist, they define inner products on the original space under which squared distances and norms encode syntax trees – even though the models being probed were never given trees as input or supervised to reconstruct them. This is a structural property of the word representation space, akin to vector offsets encoding word analogies (Mikolov et al., 2013b). Using our probe, we conduct a targeted case study, showing that ELMo (Peters et al., 2018a) and BERT (Devlin et al., 2019) representations embed parse trees with high consistency in contrast to baselines, and in a low-rank space.[1]

This chapter contributes a simple structural probe for finding syntax in word representations (§2.2), and experiments providing insights into and examples of how a low-rank transformation recovers parse trees from ELMo and BERT representations (§2.3,2.4). Finally, we discuss our probe and limitations in the context of recent work (§2.5).

2.2 Methods

Our goal is to design a simple method for testing whether a neural network embeds each sentence's dependency parse tree in its contextual word representations – a structural hypothesis. Under a reasonable definition, to embed a graph is to learn a vector representation of each node such that geometry in the vector space—distances and norms—approximates geometry in the graph (Hamilton et al., 2017). Intuitively, why do parse tree distances and depths matter to syntax? The distance metric—the path length between each pair of words—recovers the tree T simply by identifying that nodes u, v with distance $d_T(u, v) = 1$ are neighbors. The node with greater norm—depth in the tree—is the child. Beyond this identity, the distance metric explains hierarchical behavior. For example, the ability to perform the classic hierarchy test of subject-verb number agreeement (Linzen et al., 2016) in the presence of "attractors" can be explained as the verb (V) being closer in the tree to its subject (S) than to any of the attactor nouns:



Intuitively, if a neural network embeds parse trees, it likely will not use its entire representation space to do so, since it needs to encode many kinds of information. Our probe learns a linear transformation of a word representation space such that the transformed space embeds parse trees across all sentences. This can be interpreted as finding the part of the representation space that is used to encode syntax; equivalently, it is finding the distance on the original space that best fits the tree metrics.

2.2.1 The structural probe

In this section we provide a description of our proposed structural probe, first discussing the distance formulation. Let $\mathcal{M}$ be a model that takes in a sequence of n words $w^{\ell}_{1:n}$ and produces a sequence of vector representations $\mathbf{h}^{\ell}_{1:n}$, where ℓ identifies the sentence. Starting with the dot product, recall that we can define a family of inner products, $\mathbf{h}^T A\mathbf{h}$, parameterized by any positive semi-definite, symmetric matrix $A \in \mathbb{S}^{m\times m}_{+}$. Equivalently, we can view this as specifying a linear transformation $B \in \mathbb{R}^{k\times m}$, such that $A = B^T B$. The inner product is then $(B\mathbf{h})^T(B\mathbf{h})$, the norm of $\mathbf{h}$ once transformed by B. Every inner product corresponds to

[1]We release our code at `https://github.com/john-hewitt/structural-probes`.

a distance metric. Thus, our family of squared distances is defined as:

$$d_B(\mathbf{h}_i^\ell, \mathbf{h}_j^\ell)^2 = \left(B(\mathbf{h}_i^\ell - \mathbf{h}_j^\ell)\right)^T \left(B(\mathbf{h}_i^\ell - \mathbf{h}_j^\ell)\right) \tag{2.1}$$

where i, j index the word in the sentence.[2] The parameters of our probe are exactly the matrix B, which we train to recreate the tree distance between all pairs of words (w_i^ℓ, w_j^ℓ) in all sentences T^ℓ in the training set of a parsed corpus. Specifically, we approximate through gradient descent:

$$\min_B \sum_\ell \frac{1}{|s^\ell|^2} \sum_{i,j} \left| d_{T^\ell}(w_i^\ell, w_j^\ell) - d_B(\mathbf{h}_i^\ell, \mathbf{h}_j^\ell)^2 \right|$$

where $|s^\ell|$ is the length of the sentence; we normalize by the square since each sentence has $|s^\ell|^2$ word pairs.

2.2.2 Properties of the structural probe

Because our structural probe defines a valid distance metric, we get a few nice properties for free. The simplest is that distances are guaranteed nonnegative and symmetric, which fits our probing task. Perhaps most importantly, the probe tests the concrete claim that there exists an inner product on the representation space whose squared distance—a global property of the space—encodes syntax tree distance. This means that the model not only encodes which word is governed by which other word, but each word's proximity to every other word in the syntax tree.[3] This is a claim about the structure of the representation space, akin to the claim that analogies are encoded as vector-offsets in uncontextualized word embeddings (Mikolov et al., 2013b). One benefit of this is the ability to query the nature of this structure: for example, the dimensionality of the transformed space (§ 2.4.1).

2.2.3 Tree-depth structural probes

The second tree property we consider is the *parse depth* $\|w_i\|$ of a word w_i, defined as the number of edges in the parse tree between w_i and the root of the tree. This property is naturally represented as a norm – it imposes a total order on the words in the sentence. We wish to probe to see if there exists a squared norm on the word representation space that encodes this tree norm. We replace the vector distance function $d_B(\mathbf{h}_i, \mathbf{h}_j)$ with the squared vector norm $\|\mathbf{h}_i\|_B^2$, replacing Equation 2.1 with $\|\mathbf{h}_i\|_A = (B\mathbf{h}_i)^T(B\mathbf{h}_i)$ and training B to recreate $\|w_i\|$. Like the distance probe, this norm formulation makes a concrete claim about the structure of the vector space.

[2] As noted in Eqn 2.1, in practice, we find that approximating the parse tree distance and norms with the *squared* vector distances and norms consistently performs better. Because a distance metric and its square encode exactly the same parse trees, we use the squared distance throughout this paper. Also strictly, since A is not positive definite, the inner product is indefinite, and the distance a pseudometric. Further discussion can be found in our appendix.

[3] Probing for distance instead of headedness also helps avoid somewhat arbitrary decisions regarding PP headedness, the DP hypothesis, and auxiliaries, letting the representation "disagree" on these while still encoding roughly the same global structure. See Section 2.5 for more discussion.

	Distance		Depth	
Method	UUAS	DSpr.	Root%	NSpr.
LINEAR	48.9	0.58	2.9	0.27
ELMO0	26.8	0.44	54.3	0.56
DECAY0	51.7	0.61	54.3	0.56
PROJ0	59.8	0.73	64.4	0.75
ELMO1	77.0	0.83	86.5	0.87
BERTBASE7	79.8	0.85	88.0	0.87
BERTLARGE15	**82.5**	0.86	89.4	0.88
BERTLARGE16	81.7	**0.87**	**90.1**	**0.89**

Table 2.1: Results of structural probes on the PTB WSJ test set; baselines in the top half, models hypothesized to encode syntax in the bottom half. For the distance probes, we show the Undirected Unlabeled Attachment Score (UUAS) as well as the average Spearman correlation of true to predicted distances, DSpr. For the norm probes, we show the root prediction accuracy and the average Spearman correlation of true to predicted norms, NSpr.

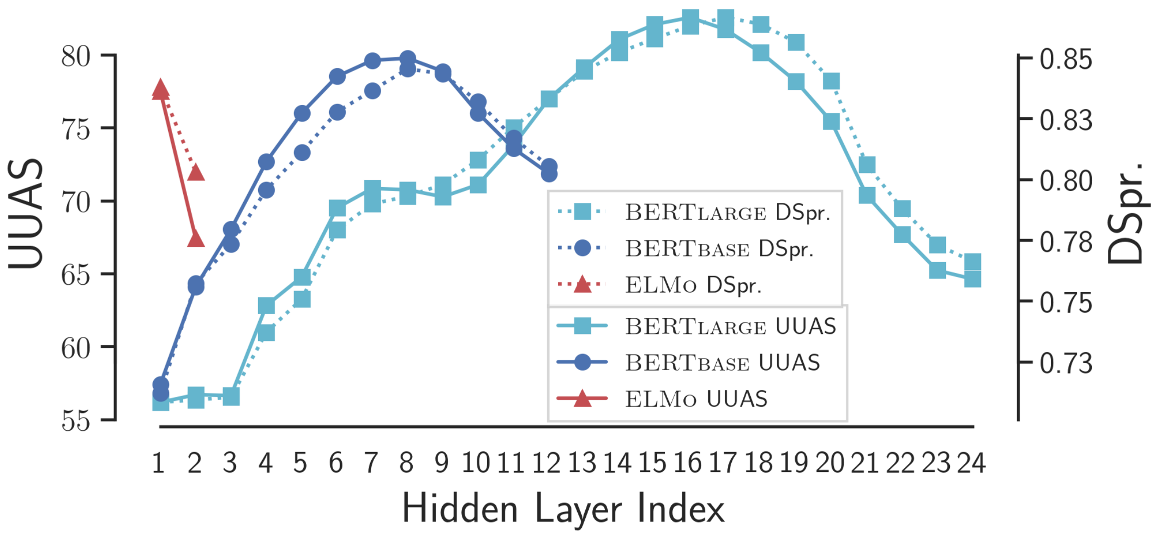


Figure 2.1: Parse distance UUAS and distance Spearman correlation across the BERT and ELMo model layers.

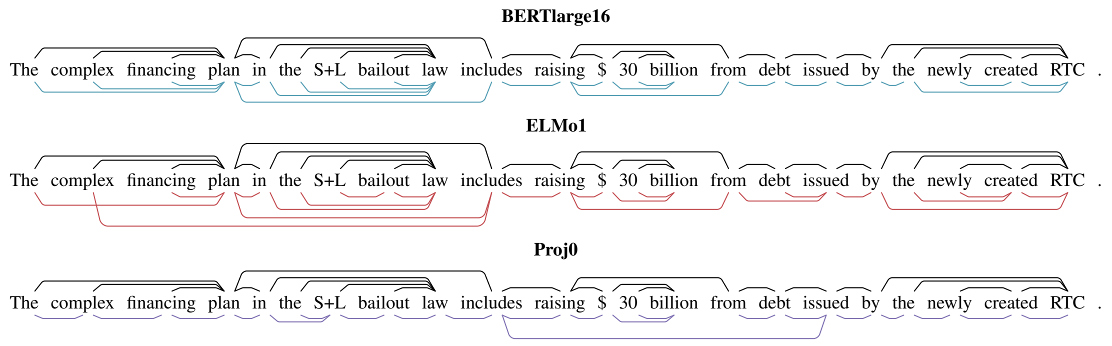

BERTlarge16
The complex financing plan in the S+L bailout law includes raising $ 30 billion from debt issued by the newly created RTC .
ELMo1
The complex financing plan in the S+L bailout law includes raising $ 30 billion from debt issued by the newly created RTC .
Proj0
The complex financing plan in the S+L bailout law includes raising $ 30 billion from debt issued by the newly created RTC .

Figure 2.2: Minimum spanning trees resultant from predicted squared distances on BERTLARGE16 and ELMO1 compared to the best baseline, PROJ0. Black edges are the gold parse, above each sentence; blue are BERTLARGE16, red are ELMO1, and purple are PROJ0.

2.3 Experiments

Using our probe, we evaluate whether representations from ELMo and BERT, two popular English models pre-trained on language modeling-like objectives, embed parse trees according to our structural hypothesis. Unless otherwise specified, we permit the linear transformation B to be potentially full-rank (i.e., B is square.) Later, we explore what rank of transformation is actually necessary for encoding syntax (§ 2.4.1).

Representation models We use the 5.5B-word pre-trained ELMo weights for all ELMo representations, and both BERT-base and BERT-large. The representations we evaluate are denoted ELMOK, BERTBASEK, BERTLARGEK, where K indexes the hidden layer of the corresponding model. All ELMo and BERT-large layers are dimensionality 1024; BERT-base layers are dimensionality 768.

Data We probe models for their ability to capture the Stanford Dependencies formalism (de Marneffe et al., 2006), claiming that capturing most aspects of the formalism implies an understanding of English syntactic structure. To this end, we obtain fixed word representations for sentences of the parsing train/dev/test splits of the Penn Treebank (Marcus et al., 1993), with no pre-processing.[4]

Baselines Our baselines should encode features useful for training a parser, but not be capable of parsing themselves, to provide points of comparison against ELMo and BERT. They are as follows:

LINEAR : The tree resulting from the assumption that English parse trees form a left-to-right chain. A model that encodes the positions of words should be able to meet this baseline.

ELMO0 : Strong character-level word embeddings with no contextual information. As these representations lack even position information, we should be completely unable to find syntax trees embedded.

[4]Since BERT constructs subword representations, we align subword vectors with gold Penn Treebank tokens, and assign each token the average of its subword representation. This thus represents a lower-bound on BERT's performance.

DECAY0 : Assigns each word a weighted average of all ELMO0 embeddings in the sentence. The weight assigned to each word decays exponentially as $\frac{1}{2^d}$, where d is the linear distance between the words.

PROJ0 : Contextualizes the ELMO0 embeddings with a randomly initialized BiLSTM layer of dimensionality identical to ELMo (1024), a surprisingly strong baseline for contextualization (Conneau et al., 2018).

2.3.1 Tree distance evaluation metrics

We evaluate models on how well the predicted distances between all pairs of words reconstruct gold parse trees and correlate with the parse trees' distance metrics. To evaluate tree reconstruction, we take each test sentence's predicted parse tree distances and compute the minimum spanning tree. We evaluate the predicted tree on undirected attachment score (UUAS)—the percent of undirected edges placed correctly—against the gold tree. For distance correlation, we compute the Spearman correlation between true and predicted distances for each word in each sentence. We average these correlations between all sentences of a fixed length, and report the macro average across sentence lengths 5–50 as the "distance Spearman (DSpr.)" metric.[5]

2.3.2 Tree depth evaluation metrics

We evaluate models on their ability to recreate the order of words specified by their depth in the parse tree. We report the Spearman correlation betwen the true depth ordering and the predicted ordering, averaging first between sentences of the same length, and then across sentence lengths 5–50, as the "norm Spearman (NSpr.)". We also evaluate models' ability to identify the root of the sentence as the least deep, as the "root%".[6]

2.4 Results

We report the results of parse distance probes and parse depth probes in Table 2.1. We first confirm that our probe can't simply "learn to parse" on top of any informative representation, unlike parser-based probes (Peters et al., 2018b). In particular, ELMO0 and DECAY0 fail to substantially outperform a right-branching-tree oracle that encodes the linear sequence of words. PROJ0, which has all of the representational capacity of ELMO1 but none of the training, performs the best among the baselines. Upon inspection, we found that our probe on PROJ0 improves over the linear hypothesis with mostly simple deviations from linearity, as visualized in Figure 2.2.

We find surprisingly robust syntax embedded in each of ELMo and BERT according to our probes. Figure 2.2 shows the surprising extent to which a minimum spanning tree on predicted distances recovers the dependency parse structure in both ELMo and BERT. As we note however, the distance metric itself is a global notion; all pairs of words are trained to know their distance – not just which word is their head; Figure 2.4 demonstrates the rich structure of the true parse distance metric recovered by the predicted distances.

[5]The 5–50 range is chosen to avoid simple short sentences as well as sentences so long as to be rare in the test data.

[6]In UUAS and "root%" evaluations, we ignore all punctuation tokens, as is standard.

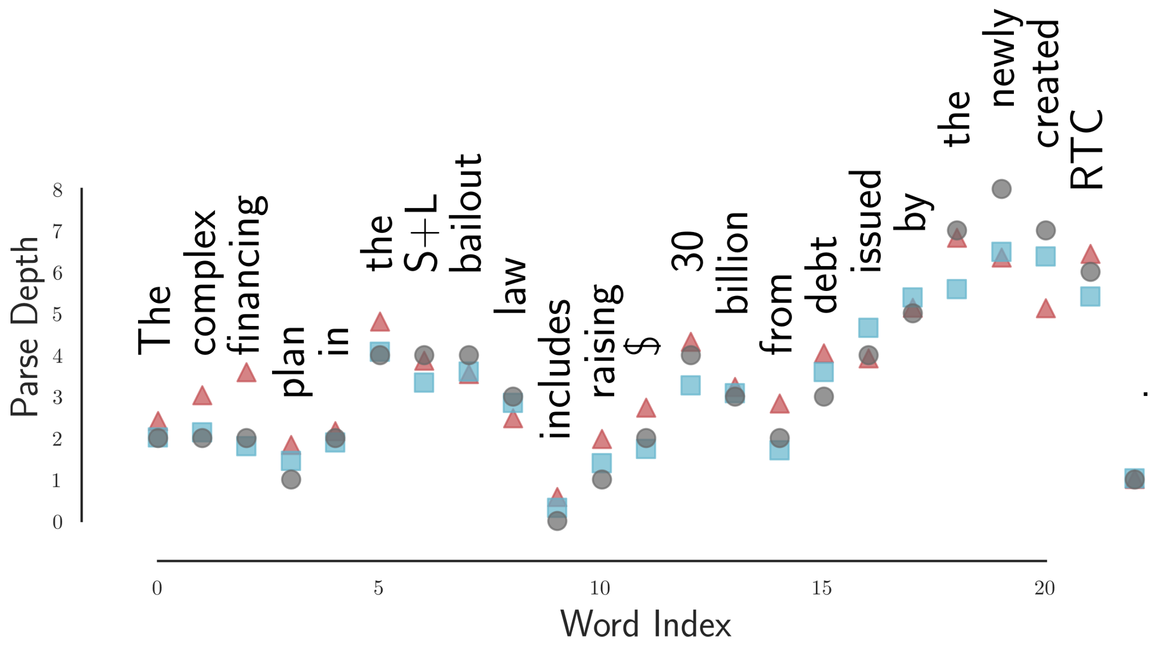


Figure 2.3: Parse tree depth according to the gold tree (black, circle) and the norm probes (squared) on ELMo1 (red, triangle) and BERTLARGE16 (blue, square).

Figure 2.3 demonstrates the surprising extent to which the depth in the tree is encoded by vector norm after the probe transformation. Between models, we find consistently that BERTLARGE performs better than BERTBASE, which performs better than ELMo.[7] We also find, as in Peters et al. (2018b), a clear difference in syntactic information between layers; Figure 2.1 reports the performance of probes trained on each layer of each system.

2.4.1 Analysis of linear transformation rank

With the result that there exists syntax-encoding vector structure in both ELMo and BERT, it is natural to ask how compactly syntactic information is encoded in the vector space. We find that in both models, the effective rank of linear transformation required is surprisingly low. We train structural probes of varying k, that is, specifying a matrix $B \in \mathbb{R}^{k \times m}$ such that the transformed vector $B\mathbf{h}$ is in $\mathbb{R}^k$. As shown in Figure 2.5, increasing k beyond 64 or 128 leads to no further gains in parsing accuracy. Intuitively, larger k means a more expressive probing model, and a larger fraction of the representational capacity of the model being devoted to syntax. We also note with curiosity that the three models we consider all seem to require transformations of approximately the same rank; we leave exploration of this to exciting future work.

[7]It is worthwhile to note that our hypotheses were developed while analyzing LSTM models like ELMo, and applied without modification on the self-attention based BERT models.

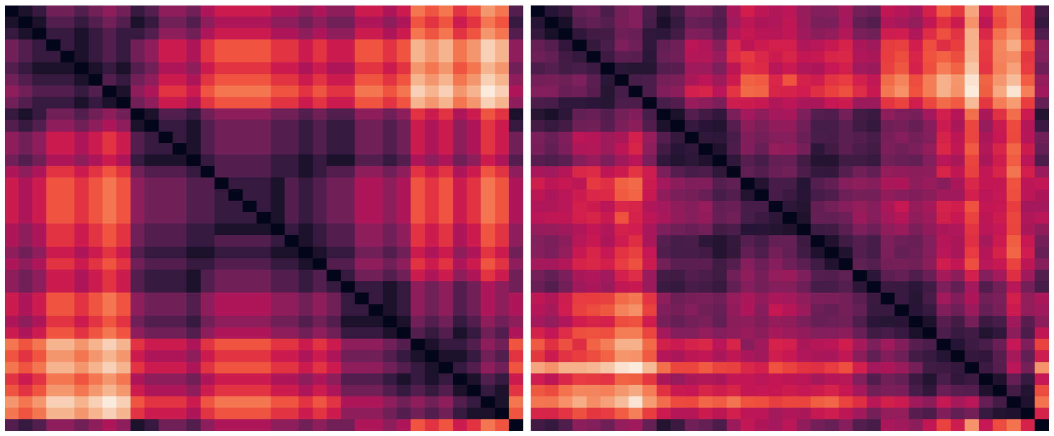


Figure 2.4: (left) Matrix representing gold tree distances between all pairs of words in a sentence, whose linear order runs top-to-bottom and left-to-right. Darker colors indicate close words, lighter indicate far. (right) The same distances as embedded by BERTLARGE16 (squared). More detailed graphs available in the Appendix.

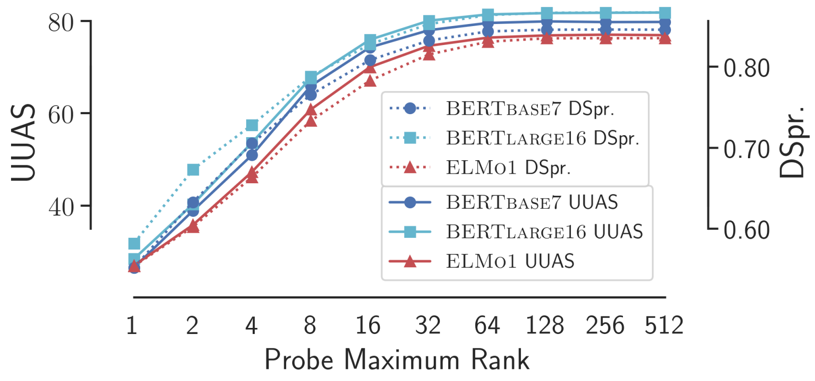


Figure 2.5: Parse distance tree reconstruction accuracy when the linear transformation is constrained to varying maximum dimensionality.

2.5 Discussion & Conclusion

Recent work has analyzed model behavior to determine if a model understands hierarchy and other linguistic phenomena (Linzen, 2018; Gulordava et al., 2018; Kuncoro et al., 2018; Linzen and Leonard, 2018; van Schijndel and Linzen, 2018; Tang et al., 2018; Futrell et al., 2018). Our work extends the literature on linguistic probes, found at least in (Peters et al., 2018b; Belinkov et al., 2017; Blevins et al., 2018; Hupkes et al., 2018). Conneau et al. (2018) present a task similar to our parse depth prediction, where a sentence representation vector is asked to classify the maximum parse depth ever achieved in the sentence. Tenney et al. (2019) evaluates a complementary task to ours, training probes to learn the *labels* on structures when the gold structures themselves are given. Peters et al. (2018b) evaluates the extent to which constituency trees can be extracted from hidden states, but uses a probe of considerable complexity, making less concrete hypotheses about how the information is encoded.

Probing tasks and limitations. Our reviewers rightfully noted that one might just probe for headedness, as in a bilinear graph-based dependency parser. More broadly, a deep neural network probe of some kind is almost certain to achieve higher parsing accuracies than our method. Our task and probe construction are designed not to test for some notion of syntactic knowledge broadly construed, but instead for an extremely strict notion where all pairs of words know their syntactic distance, and this information is a global structural property of the vector space. However, this study is limited to testing that hypothesis, and we foresee future probing tasks which make other tradeoffs between probe complexity, probe task, and hypotheses tested.

In summary, through our structural probes we demonstrate that the structure of syntax trees emerges through properly defined distances and norms on two deep models' word representation spaces. Beyond this actionable insight, we suggest our probe may be useful for testing the existence of different types of graph structures on any neural representation of language, an exciting avenue for future work.

Later work. In the years since the work presented in this chapter was published as Hewitt and Manning (2019), considerable work has followed up on, examined, confirmed, and questioned the methodology. Reif et al. (2019) discussed why squared L2 distance (as we use) works better than the standard L2 distance, as L2 distance cannot embed tree metrics without distortion; however, this did not explain the core curiousity of mine, which was why squared L2 distance works better than L1 distance, which is the natural tree distance metric (Deza and Laurent, 2009). Chi et al. (2020) found that our results hold across languages, and in fact training a probe on a multilingual language model using supervision from one natural language can extract syntax in other languages, suggesting that the syntactic representations are shared across languages. Many variants of the structural probe have been proposed, including changing the functional form to be more expressive (White et al., 2021) or less (Limisiewicz and Mareček, 2021). Overall, the trends we saw seem to hold across a range of networks.

2.6 Implementation Details

2.6.1 Squared L2 distance vs. L2 distance

In Section 2.2.2, we note that while our distance probe specifies a distance metric, we recreate it with a squared vector distance; likewise, while our norm probe specifies a norm, we recreate it with a squared vector norm. We found this to be important for recreating the exact parse tree distances and norms. This does mean that in order to recreate the exact scalar values of the parse tree structures, we need to use the squared vector quantities. This may be problematic, since for example squared distance doesn't obey the triangle inequality, whereas a valid distance metric does.

However, we note that in terms of the graph structures encoded, distance and squared distance are identical. After training with the squared vector distance, we can square-root the predicted quantities to achieve a distance metric. The relative ordering between all pairs of words will be unchanged; the same tree is encoded either way, and none of our quantitative metrics will change; however, the exact scalar distances will differ from the true tree distances.

(Reif et al., 2019) give an argument as to why squared distance works better than L2 distance: squared L2 distance can embed tree distances without distortion; L2 cannot. Why squared L2 works better than L1 distance, which also can embed tree distances without distortion and yet is a valid metric, remains an open question.

2.6.2 Probe training details

All probes are trained to minimize L1 loss of the predicted squared distance or squared norm w.r.t. the true distance or norm. Optimization is performed using the Adam optimizer (Kingma and Ba, 2014) initialized at learning rate 0.001, with $\beta_1 = .9, \beta_2 = .999, \epsilon = 10^{-8}$. Probes are trained to convergence, up to 40 epochs, with a batch size of 20. For depth probes, loss is summed over all predictions in a sentence, normalized by the length of the sentence, and then summed over all sentences in a batch before a gradient step is taken. For distance probes, normalization is performed by the square of the length of the sentence. At each epoch, dev loss is computed; if the dev loss does not achieve a new minimum, the optimizer is reset (no momentum terms are kept) with an initial learning rate multiplied by 0.1. All models were implemented in both DyNet (Neubig et al., 2017), and in PyTorch (Paszke et al., 2017).

2.7 Extra examples

In this section we provide additional examples of model behavior, including baseline model behavior, across parse distance prediction and parse depth prediction. In Figure 2.6 and Figure 2.7, we present a single sentence with dependency trees as extracted from many of our models and baselines. In Figure 2.8, we present tree depth predictions on a complex sentence from ELMO1, BERTLARGE16, and our baseline PROJ0. Finally, in

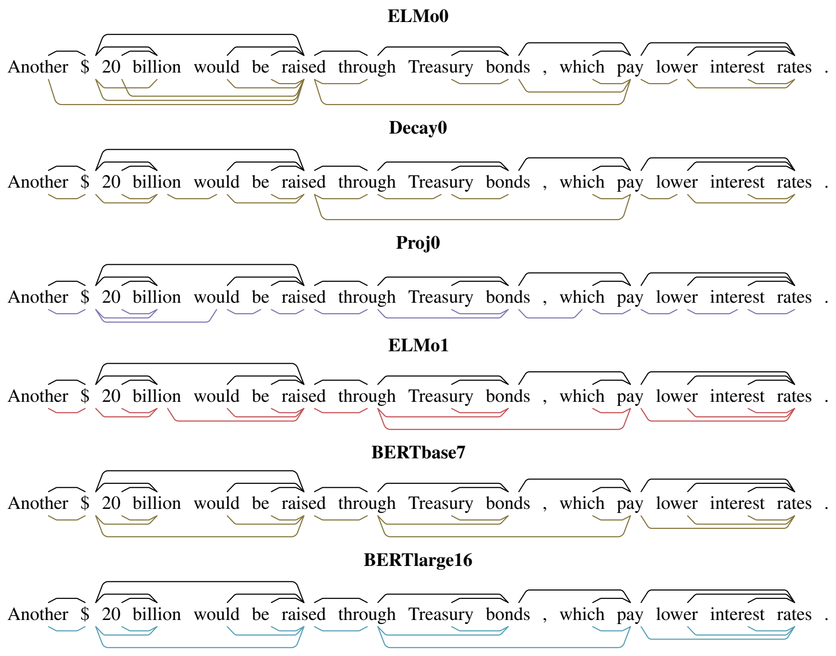

ELMo0
Decay0
Proj0
ELMo1
BERTbase7
BERTlarge16
Another $ 20 billion would be raised through Treasury bonds , which pay lower interest rates .

Figure 2.6: A relatively simple sentence, and the minimum spanning trees extracted by various models. Black edges correspond to the gold (true) parse trees.

Figure 2.9, we present gold parse distances and predicted squared parse distances between all pairs of words in large, high-resolution format.

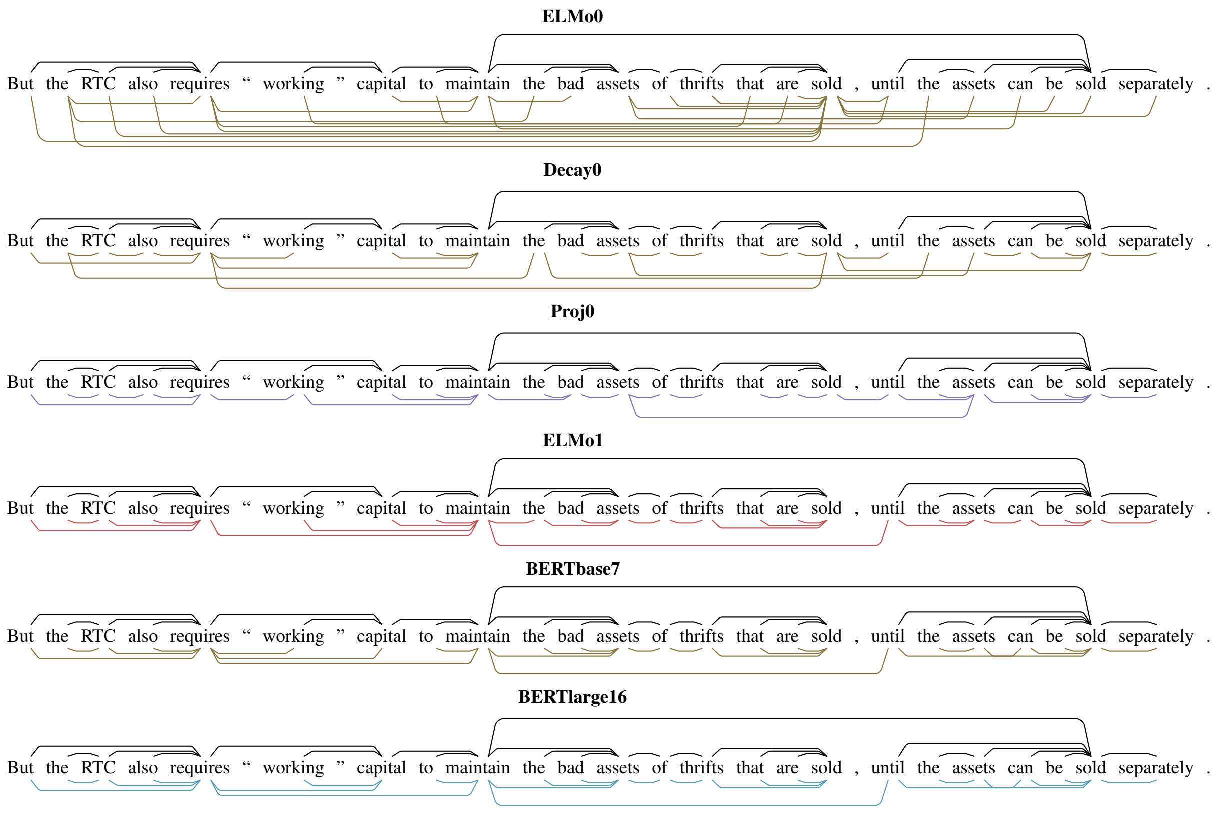

ELMo0
Decay0
Proj0
ELMo1
BERTbase7
BERTlarge16
But the RTC also requires “ working ” capital to maintain the bad assets of thrifts that are sold , until the assets can be sold separately .

Figure 2.7: A complex sentence, and the minimum spanning trees extracted by various models.

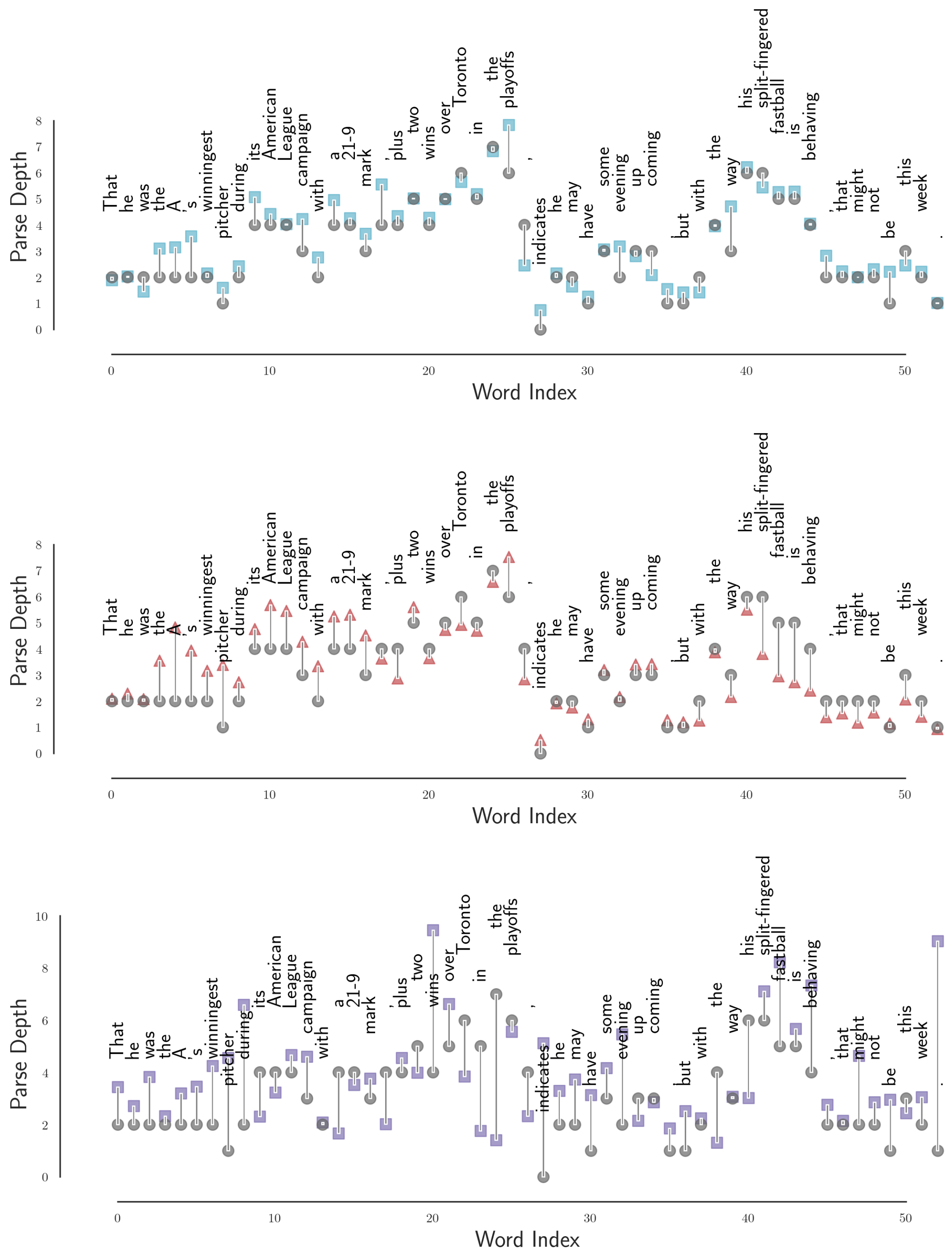


Figure 2.8: A long sentence with gold dependency parse depths (grey) and dependency parse depths (squared) as extracted by BERTLARGE16 (blue, top), ELMO1 (red, middle), and the baseline PROJ0 (purple, bottom). Note the non-standard subject, “that he was the A’s winningest pitcher".

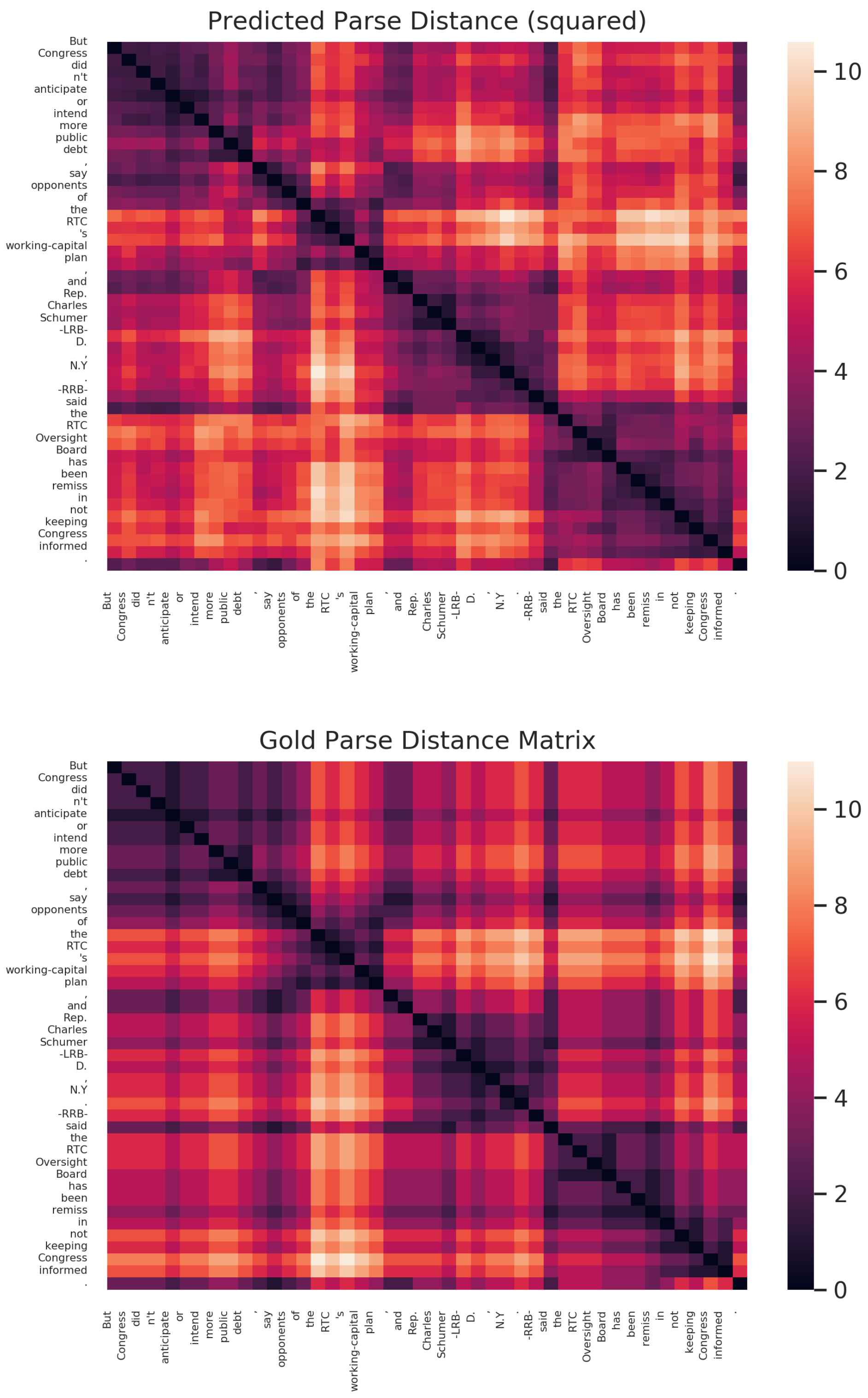


Figure 2.9: The distance graphs defined by the gold parse distances on a sentence (below) and as extracted from BERTLARGE16 (above, squared).

Chapter 3

Designing and Interpreting Probes with Control Tasks

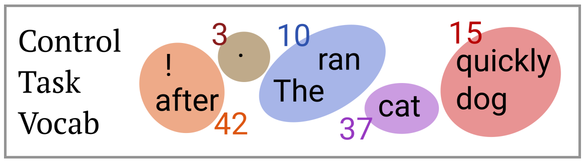
3.1 Introduction

In Chapter 2, we demonstrated using probing methods that there are simple functions that extract syntactic trees from neural language models like BERT. Separately, probes trained on various representations have obtained high accuracy on tasks requiring part-of-speech and morphological information (Belinkov et al., 2017), syntactic and semantic information (Peters et al., 2018b; Tenney et al., 2019), among other properties (Conneau et al., 2018), providing evidence that deep representations trained on large datasets are predictive of a broad range of linguistic properties.

But when a probe achieves high accuracy on a linguistic task using a representation, can we conclude that the representation encodes linguistic structure, or has the probe just learned the task? Probing papers tend to acknowledge this uncertainty, putting accuracies in context using random representation baselines (Zhang and Bowman, 2018) and careful task design (Hupkes et al., 2018). Even so, as long as a representation is a lossless encoding, a sufficiently expressive probe with enough training data can learn *any* task on top of it.

In this chapter, we propose *control tasks*, which associate word types with random outputs, to give intuition for the expressivity of probe families and provide insight into how representation and probe interact to achieve high task accuracy.

Control tasks are based on the intuition that the more a probe is able to make task output decisions independently of the linguistic properties of a representation, the less its accuracy on a linguistic task necessarily reflects the properties of the representation. Thus, a good probe (one that provides insights into the linguistic properties of a representation) should be what we call *selective*, achieving high linguistic task accuracy and low control task accuracy (see Figure 3.2).



Sentence 1	The	cat	ran	quickly	.
Part-of-speech	DT	NN	VBD	RB	.
Control task	**10**	**37**	**10**	**15**	**3**
Sentence 2	The	dog	ran	after	!
Part-of-speech	DT	NN	VBD	IN	.
Control task	**10**	**15**	**10**	**42**	**42**

Figure 3.1: Our control tasks define random behavior (like a random output, top) for each word type in the vocabulary. Each word token is assigned its type's output, regardless of context (middle, bottom.) Control tasks have the same input and output space as a linguistic task (e.g., parts-of-speech) but can only be learned if the probe memorizes the mapping.

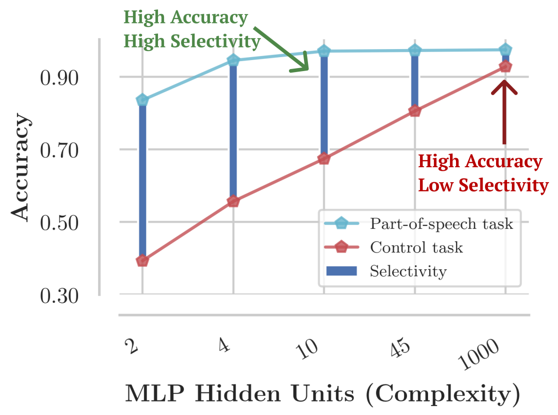


Figure 3.2: Selectivity is defined as the difference between linguistic task accuracy and control task accuracy, and can vary widely, as shown, across probes which achieve similar linguistic task accuracies. These results taken from § 3.3.5.

We show that selectivity can be a guide in designing probes and interpreting probing results, complementary to random representation baselines; as of now, there is little consensus on how to design probes. Early probing papers used linear functions (Shi et al., 2016; Ettinger et al., 2016; Alain and Bengio, 2016), which are still used (Bisazza and Tump, 2018; Liu et al., 2019a), but multi-layer perceptron (MLP) probes are at least as popular (Belinkov et al., 2017; Conneau et al., 2018; Adi et al., 2017; Tenney et al., 2019; Ettinger et al., 2018). Arguments have been made for "simple" probes, e.g., that we want to find easily accessible information in a representation (Liu et al., 2019a; Alain and Bengio, 2016). As a counterpoint though, "complex" MLP probes have also been suggested since useful properties might be encoded non-linearly (Conneau et al., 2018), and they tend to report similar trends to simpler probes anyway (Belinkov et al., 2017; Qian et al., 2016).

We define control tasks corresponding to English part-of-speech tagging and dependency edge prediction, and use ELMo representations to conduct a broad study of probe families, hyperparameters, and regularization

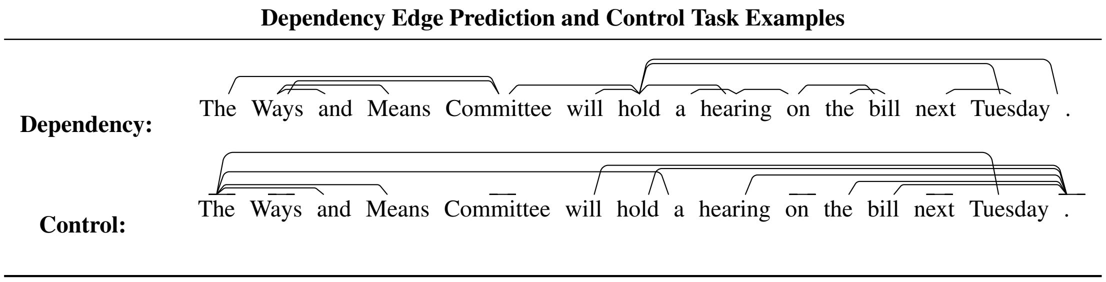

Dependency Edge Prediction and Control Task Examples
Dependency:
The Ways and Means Committee will hold a hearing on the bill next Tuesday .
Control:
The Ways and Means Committee will hold a hearing on the bill next Tuesday .

Figure 3.3: Example dependency tree from the development set of the Penn Treebank with dependents pointing at heads, and the structure resulting from our dependency edge prediction control task on the same sentence.

methods, evaluating both linguistic task accuracy and selectivity. We propose that selectivity be used for building intuition about the expressivity of probes and the properties of models, putting probing accuracies into richer context. We find that:

1. With popular hyperparameter settings, MLP probes achieve very low selectivity, suggesting caution in interpreting how their results reflect properties of representations. For example, on part-of-speech tagging, 97.3 accuracy is achieved, compared to 92.8 control task accuracy, resulting in 4.5 selectivity.
2. Linear and bilinear probes achieve relatively high selectivity across a range of hyperparameters. For example, a linear probe on part-of-speech tagging achieves a similar 97.2 accuracy, and 71.2 control task accuracy, for 26.0 selectivity. This suggests that the small accuracy gain of the MLP may be explained by increased probe expressivity.
3. The most popular method for controlling probe complexity, dropout, does not consistently lead to selective MLP probes. However, control of MLP complexity through unintuitively small (10-dimensional) hidden states, as well as small training sample sizes and weight decay, lead to higher selectivity and similar linguistic task accuracy.

Finally, we ask, *can we meaningfully compare the linguistic properties of layers of a model using only linguistic task accuracy*? We raise a potential problem with this approach: it fails to take into account differences in ease of memorization across layers. In particular, we find that while linear and MLP probes on the first layer of ELMo (ELMo1) achieve slightly higher part-of-speech accuracy than those on the second layer (ELMo2), (97.2 compared to 96.6, for a loss of 0.6), the same probes achieve much greater selectivity on ELMo2 (31.4 compared to 26.0, for a gain of 5.4). Thus, the difference in selectivity in favor of ELMo2 is much greater than the commonly known (Peters et al., 2018a; Liu et al., 2019a) difference in linguistic task accuracy in favor of ELMo1; the difference in accuracy may be explained by probes more easily accessing word identity features in ELMo1.

3.2 Control Tasks

In this section, we describe how to construct control tasks. At a high level, control tasks have:

structure: The output for a word token is a deterministic function of the word type[1].

randomness: The output for each word type is sampled independently at random.

We start with some notation; denote as $1 : T$ the sequence of integers $\{1, ..., T\}$. Let V be the vocabulary containing all word types in a corpus. A sentence of length T is $\mathbf{x}_{1:T}$, where each $x_i \in V$, and the word representations of the model being probed are $\mathbf{h}_{1:T}$, where $h_i \in \mathbb{R}^d$. A *task* is a function that maps a sentence to a single output per word, $f(\mathbf{x}_{1:T}) = \mathbf{y}_{1:T}$, where each output is from a finite set of outputs: $y_i \in \mathcal{Y}$. Each *control task* is defined in reference to a *linguistic task*, and the two share $\mathcal{Y}$. We'll now use part-of-speech tagging and dependency edge prediction as examples to describe the construction of control tasks.

3.2.1 Part-of-speech tagging control task

In part-of-speech tagging, the set $\mathcal{Y}$ is the tagset, $1 : 45$ (corresponding to NN, NNS, VB,...). To construct a control task, we independently sample a *control behavior* $C(v)$ for each $v \in V$. The control behavior specifies how to define $y_i \in \mathcal{Y}$ for a word token x_i with word type v. For part-of-speech tagging, each control behavior directly specifies the output y_i for x_i as an integer from $1 : 45$, so we sample from 45 behaviors[2]. The part-of-speech control task is the function that maps each token x_i to the label specified by the behavior $C(x_i)$:

$$f_{\text{control}}(\mathbf{x}_{1:T}) = f(C(x_1), C(x_2), ...C(x_T)). \tag{3.1}$$

This task is visualized in Figure 3.1.

3.2.2 Dependency edge prediction control task

The dependency edge prediction task is the function $f_{\text{DEP}}(\mathbf{x}_{1:T}) = \mathbf{y}_{1:T}$ where y_i is the index of the parent of x_i in the dependency tree on the sentence $\mathbf{x}_{1:T}$. Thus, the output space $\mathcal{Y} = 1 : T$ depends on the length of the sentence, T. To accommodate this in our control task, we define the control behaviors $C(v)$ in a length-independent way that still fully specifies y_i. The possible behaviors $C(v)$ are as follows:

attach to self: Always attach tokens of this type to themselves. That is, $y_i = i$.

attach to first: Always attach tokens of this type to the first token. That is, $y_i = 1$.

attach to last: Always attach tokens of this type to the last word in the sentence. That is, $y_i = T$.

We sample uniformly from the three. Given these behaviors, the control task is defined as before by Eqn 3.1. This task is visualized in Figure 3.3.

While very similar to dependency parsing, dependency edge prediction differs in two ways. The output is not constrained to be a tree for evaluation; each prediction is evaluated independently. So, while our control

[1]Equivalently, word identity.

[2]The exact distribution from which we sample isn't crucial, but for part-of-speech tagging, we sample from the empirical token distribution of part-of-speech tagging, so the marginal probability of each label is similar.

tasks do not define trees, the two tasks' output spaces are still the same. Second, in dependency edge prediction, the root of the sentence is omitted from evaluation; no sentence-external `ROOT` token is posited for evaluation.

3.2.3 Properties of control tasks

To summarize, a control task is defined for a single linguistic task, and shares the linguistic task's output space $\mathcal{Y}$. To construct a control task, a control behavior $C(v)$ is sampled independently at random for each word type $v \in V$. The control task is a function mapping $\mathbf{x}_{1:T}$ to a sequence of outputs $\mathbf{y}_{1:T}$ which is fully specified by the sequence of behaviors, $[C(x_1), ..., C(x_T)]$.

From this construction, we note that the ceiling on performance is the fraction of tokens in the evaluation set whose types occur in the training set (plus chance accuracy on all other tokens.) Further, $C(v)$ must be memorized independently for each word type, and a probe taking vectors $h_{1:T}$ as input must identify for each h_i its corresponding x_i, and output the element of $\mathcal{Y}$ specified by $C(x_i)$.

3.3 Experiments on Probe Selectivity

In this section, we conduct a broad study of probe families (e.g, linear, MLP) and hyperparameter choices (weight matrix rank/hidden state size, amount of regularization) on a single representation (ELMo1) to determine (1) what probe choices exhibit high linguistic task accuracy *and* high selectivity (and whether this holds for a range of hyperparameters), and (2) whether each probe family can be made selective through hyperparameter choices without substantially sacrificing linguistic task accuracy.

3.3.1 Probe families

We experiment with three types of probes per task.

For part-of-speech tagging, we experiment with linear, MLP-1, and MLP-2 probes. The linear probe is a multiclass model mapping h_i to $y_i \sim \text{softmax}(Ah_i + b)$. The MLP-1 probe is a multilayer perceptron with one hidden layer and ReLU nonlinearity defined as:

$$y_i \sim \text{softmax}(W_2\, g(W_1 h_i)). \tag{3.2}$$

And the MLP-2 probe is defined as:

$$y_i \sim \text{softmax}(W_3\, g(W_2\, g(W_1 h_i))). \tag{3.3}$$

where g is the ReLU function, and bias terms are omitted from all affine transformations for brevity.

For dependency edge prediction, we experiment with bilinear, MLP-1, and MLP-2 probes. These probes take as input the entire sequence $\mathbf{h}_{1:T}$ as well as the vector h_i of a given state to produce y_i; the softmax operates over the sequence to construct a distribution over the T classes. Formally, the bilinear model is defined as $y_i \sim \text{softmax}(\mathbf{h}_{1:T}^\top A h_i + b)$. The MLP-1 probe is defined as follows:

$$y_i \sim \text{softmax}(W_2\ g(W_1[\mathbf{h}_{1:T}; h_i])). \tag{3.4}$$

Note here that h_i broadcasts to $\mathbb{R}^{T\times d}$,while $W_1 \in \mathbb{R}^{\ell\times d}$, and $W_2 \in \mathbb{R}^{1\times\ell}$ broadcast as well. That is, each $[h_j; h_i]$ pair is mapped to a single scalar independently of all others, leading to T logits used as input to the softmax. Similarly, the MLP-2 model is defined as follows:

$$y_i \sim \text{softmax}(W_3\ g(W_2\ g(W_1[\mathbf{h}_{1:T}; h_i]))). \tag{3.5}$$

3.3.2 Complexity control

It is well-known that probes should not be too complex (Liu et al., 2019a; Alain and Bengio, 2016); this is the motivation behind constraining the input to the probe to be a single vector or pair of vectors. However, there has been no systematic investigation of probe complexity. We study what complexity control is necessary to achieve selectivity. As we will see, the typical practice of regularizing to reduce the generalization gap (difference between training and test task accuracy) is insufficient if one is interested in selectivity.

Rank/hidden dimensionality constraint. For our linear and bilinear probes, we constrain the rank of weight matrices through an LR decomposition. We let $A \in \mathbb{R}^{k\times d}$, where k is the output space (45 for part-of-speech tagging; 1 for dependency head prediction). To constrain A to rank ℓ, we factor $A = LR$, where $L \in \mathbb{R}^{k\times\ell}$ and $R \in \mathbb{R}^{\ell\times|V|}$, and optimize over L and R. For MLP models, we let the hidden state size be equal to ℓ.[3]

From the default value of rank-1000 and 1000-dimensional hidden states, we let ℓ take on the values $\{2, 4, 10, 45\}$ for part-of-speech, and $\{5, 10, 50, 100\}$ for dependency edge prediction.[4]

Dropout. We apply dropout (Srivastava et al., 2014) with probability p to the input for linear and bilinear probes, and to the input and the output of each hidden layer for MLP probes. From the default value of 0, we let p range over $\{0.2, 0.4, 0.6, 0.8\}$.

Number of training examples. We artificially constrain the number of sentences the probe is trained on, with the intuition that general rules can be learned more sample-efficiently than memorization. Zhang and Bowman (2018) showed this to be an effective distinguishing factor between trained representations and random representation controls. From the default of 39832 (the number of training examples in the dataset), we train on $\{4000, 400\}$ examples, corresponding to roughly 100%, 10%, and 1% of the total data, as suggested by Zhang and Bowman (2018).

[3]One could constrain the matrices of the MLP to be rank ℓ without making the hidden state smaller, but one must choose a hidden state size anyway, so we believed a study changing the hidden state size would be most informative.

[4]Note that for linear models, the rank is constrained by k regardless, since $A \in \mathbb{R}^{k\times d}$.

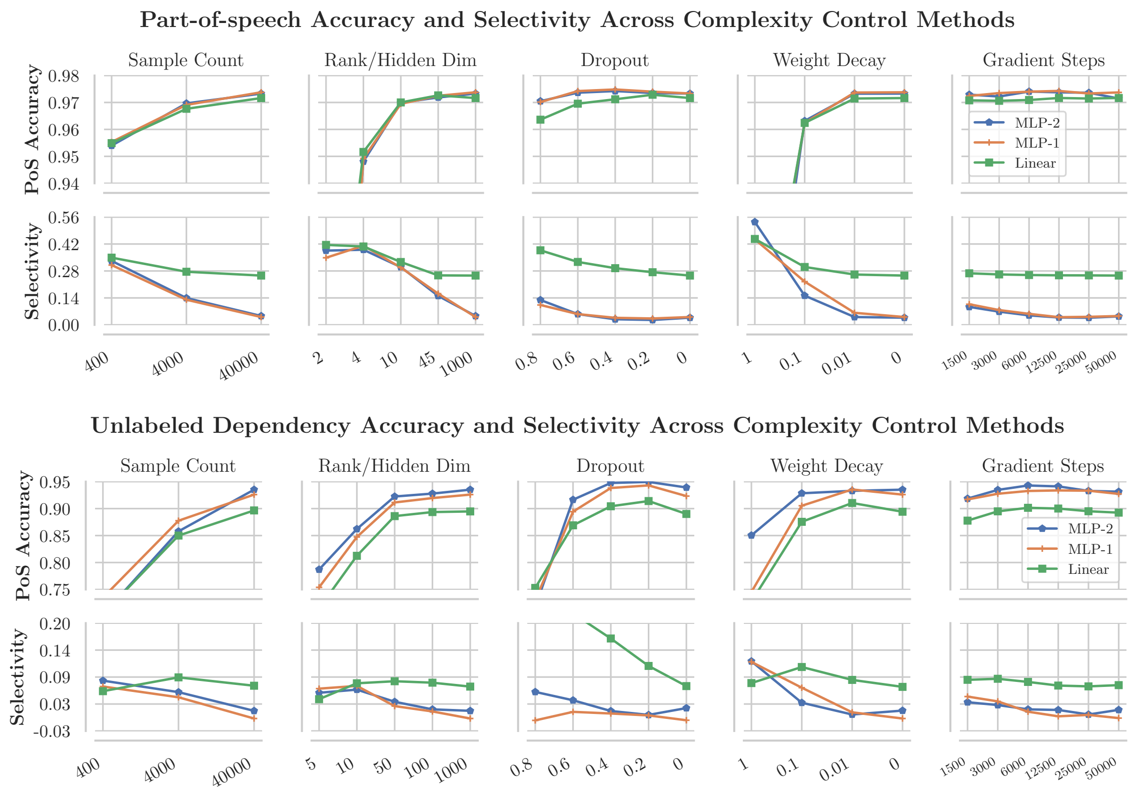


Figure 3.4: Linguistic task accuracies and selectivities for the 5 complexity control methods. All methods except dropout and early stopping are shown to improve selectivity without a large impact on linguistic task accuracy. All methods for the same task share a common y-axis, and use their own categorical x-axis. All x-axes are ordered from most severe constraints on complexity (left) to most laissez-faire (right).

L_2 **regularization.** We apply weight decay to the probe parameters. From the default of 0, we let the weight decay constant take on the values $\{0.01, 0.1, 1.0, 10.0\}$, unnormalized by batch size.

Early stopping. All of our probing models are trained with Adam (Kingma and Ba, 2014). By default, we anneal the learning rate by a factor of 0.5 each time an epoch does not lead to a new minimum loss on the development set, and stop training when 4 such epochs occur in a row. However, in early stopping, we explicitly halt training at a fixed number of gradient steps. From the default of 100000 (approximately 40 epochs), we let this maximum take on the values $\{50000, 25000, 12500, 6000, 3000, 1500\}$.

3.3.3 Dataset

We use the Penn Treebank (PTB) dataset (Marcus et al., 1993) with the traditional parsing training/development/testing splits[5] without preprocessing. We report accuracies on the development set. We convert the PTB constituency trees to the Stanford Dependencies formalism (de Marneffe et al., 2006) for our dependency edge

[5]As given by the code of Qi and Manning (2017) at `https://github.com/qipeng/arc-swift`.

Probe	**PoS**	**Ctl**	**Select.**	**Dep**	**Ctl**	**Select.**
Probes with Default Hyperparameters						
Linear	97.2	71.2	26.0	-	-	-
Bilinear	-	-	-	89.0	82.4	6.6
MLP-1	97.3	92.8	4.5	92.3	93.0	-0.7
MLP-2	97.3	93.2	4.2	93.9	92.0	1.9
Probes with 0.4 Dropout						
Linear	97.1	67.3	29.8	-	-	-
Bilinear	-	-	-	90.4	73.7	16.7
MLP-1	97.5	93.4	4.1	93.8	93.1	0.7
MLP-2	97.4	94.1	3.4	94.7	93.5	1.3
Probes Designed with Control Tasks						
Linear	97.0	64.0	33.0	-	-	-
Bilinear	-	-	-	91.0	83.1	7.9
MLP-1	97.2	80.6	16.6	90.5	84.3	6.2
MLP-2	97.2	81.7	15.4	92.8	89.8	3.0

Table 3.1: Probe accuracies on linguistic tasks and control tasks. Default hyperparameters correspond to a hidden state of dimensionality 1000 and no dropout. Under Probes Designed with Control Tasks, we used selectivity to hand-pick a hyperparameter setting for each probe. In particular, part-of-speech probes designed with control tasks all use rank-10 weight matrices (10-dimensional hidden state) and no other changes. Dependency edge prediction probes designed with control tasks had, for the bilinear model, weight decay of 0.01, for MLP-1, weight decay of 0.1, for MLP-2, a rank-50 weight matrix.

prediction task.

3.3.4 Representation

We use the 5.5 billion-word pre-trained ELMo representations (Peters et al., 2018a). Since the output of the first BiLSTM layer was recently shown to be the most transferrable on a wide variety of tasks, including part-of-speech and syntax (Liu et al., 2019a), we focus on analyzing that layer, which we denote ELMo1.

3.3.5 Results

Selectivity of default hyperparameters. Our results with linear, bilinear, and MLP probes with "default" hyperparameters, as specified in § 3.3.2, are found in Table 3.1 (top). We find that linear probes achieve similar part-of-speech accuracies to MLPs (97.2 compared to 97.3) with substantially higher selectivity (26.0 vs 4.50). In dependency edge prediction, we find a definite gap between bilinear probe accuracy (89.0) and MLP-1 accuracy (92.3). However, the bilinear probe achieves 16.7 selectivity, compared to -0.7 by MLP-1 and 1.3 by MLP-2. Thus, with no regularization, modest gains in linguistic task accuracy through MLP probes over linear/bilinear probes are tempered by losses in selectivity. Bilinear and linear probes themselves show a significant capacity for memorization.

Does adding moderate regularization through dropout (e.g., $p = 0.4$) consistently lead to selectivity?

Surprisingly, as shown in Table 3.1 (middle), the opposite is true for some MLP probes, where selectivity actually decreases (e.g., $4.2 \rightarrow 3.4$ for MLP-2). In one case, the MLP-1 probe on dependency edge prediction, dropout increases selectivity (-$0.7 \rightarrow 0.7$) but for no others.

How hard is it to find selective probes? We tried 6 methods for controlling probe complexity, and all worked except dropout and early stopping, though never for a broad range of hyperparameters. For each complexity control method except dropout and early stopping, we find hyperparameters that lead to high linguistic task accuracy and high selectivity. Our results are summarized in Figure 3.4.

We find that constraining the **hidden state dimensionality** of MLPs is an effective way to encourage selectivity at little cost to linguistic task accuracy. MLP hidden state sizes of 10 and 50, for part-of-speech tagging and dependency head prediction respectively, lead to increased selectivity while maintaining high linguistic task accuracy. As such, MLP probes with hundreds or 1000 hidden units, as is common, are overparameterized.

Constraining the **number of training examples** is effective for part-of-speech, suggesting that learning each linguistic task requires fewer samples than our control task. However, for dependency edge prediction, this leads to significantly reduced linguistic task accuracy. Finally, we find that the right **weight decay** constant can also lead to high-accuracy, high-selectivity probes, especially for dependency edge prediction. As shown, however, it is unclear what hyperparameters to use (e.g., weight decay 0.1) to achieve both high accuracy and high selectivity; that is, finding selective MLP probes is non-trivial.

Applying **dropout**, the most popular probing regularization method (Adi et al., 2017; Belinkov and Glass, 2019; Şahin et al., 2019; Kim et al., 2019; Elloumi et al., 2018; Belinkov and Glass, 2017; Belinkov et al., 2018) does not consistently lead to high-accuracy, high-selectivity MLP probes across a broad range of dropout probabilities ($p = 0.2$ to $p = 0.8$) on part-of-speech tagging. For dependency edge prediction, dropout of $p = 0.6$ improves the selectivity of MLP-2 but not MLP-1, and considerably increases the already relatively large selectivity of the bilinear probe. **Early stopping** in the ranges tested also has little impact on part-of-speech tagging, selectivity, but does improve selectivity of MLP dependency edge prediction probes.

From our study, we pick a set of hyperparameters for linear, bilinear, MLP-1 and MLP-2 probes to encourage selectivity and linguistic task accuracy together, to compare to default parameters and dropout. We chose rank constraints of 10 and 45, respectively (with no other changes,) for linear and MLP part-of-speech tagging probes, weight decay of 0.01 for the bilinear dependency probe, and weight decay of 0.1 for MLP dependency probes. We report the results of these probes in Table 3.1 (bottom). In all cases, we see that the right choice of probe leads to considerably higher selectivity than dropout or no regularization. In particular, for part-of-speech tagging, our chosen MLP-1 probe achieves 16.6 selectivity, up from 4.5, and on dependency head prediction, 6.2 selectivity, up from -0.7.

3.3.6 Discussion

Our most consistent result seems to be that all probes, whether linear, bilinear, or multi-layer perceptron, are over-parameterized and needlessly high-capacity if using defaults like full-rank weight matrices, hidden states with a few hundred dimensions, and moderate dropout. We can tell this is the case because we're able to heavily constrain the probes (e.g., to rank or 10-dimensional hidden states with little loss in accuracy.

We find that the most selective probes of those tested, even after careful complexity control, are linear or bilinear models. They also have the advantage that they exhibit high selectivity without the need to search over complexity control methods.

However, the most accurate probes on the more complex task of dependency edge prediction are MLPs, even with hyperparameters tuned for selectivity. This suggests that while much of the part-of-speech information of ELMo is extractable linearly, some information about syntactic trees is not available to a bilinear function. In some cases, therefore, one might opt for an MLP probe to extract non-linear features, while optimizing for selectivity through hyperparameter choices.

Errors in Selective and Non-Selective Probes Do selective and non-selective probes make different types of errors? We ran a qualitative study on this, training ten MLP-1 probes and ten linear probes, each with default parameters, on part-of-speech tagging. We then manually inspected their aggregate confusion matrices for trends in differences between the models' errors.

While the MLP performed marginally better at recognizing many categories, the plurality of improvement over the linear probe by far was in correctly identifying the difference between nouns and adjectives in phrases. For example,

Kan.-based/JJ National/NNP Pizza/NNP
rental/JJ equipment/NN

were correctly labeled by the MLP but not the linear probe, which incorrectly labeled the adjectives as nouns. As can be seen with the second example, the distinction between a *JJ NN* modified noun and a *NN NN* noun compound is quite subtle, and the MLP picks up on the distinction considerably better.

The linear probe, however, was substantially more accurate at predicting the NNP tag, which the MLP probe frequently mislabeled as NNPS. Manual inspection showed a general trend:

Environmental/NNP Systems/NNP Co./NNP
Cara/NNP Operations/NNP Co./NNP
7.8/CD %/NN stake/NN in/IN Dataproducts/NNP

In each case, the MLP probe mislabeled the word with the suffix *-s* as NNPS. The linear probe was considerably less prone to this error. We hypothesize that this is because the MLP probe is expressive enough to pick up on (spurious) markers of plurality as well as status as a proper noun independently and combine them, whereas the linear probe is less able to do so. If this hypothesis is true, then this serves as an example of how less selective probes may be less faithful in representing the linguistic information of the model being probed, since features may be combined to make fine-grained distinctions.

3.4 Selectivity Differences Confound Layer Comparisons

In this section, we use selectivity to shed light on confounding factors when comparing the linguistic capabilities of different representations. Multiple studies have found probes on ELMo1 to perform better at part-of-speech tagging than probes on ELMo2 (Peters et al., 2018a; Tenney et al., 2019; Liu et al., 2019a). As we note, these results depend on the probe as well as the representation; given what we know about probes' capacity for memorizing at the type level, we explore an alternative to the hypothesis that ELMo1 has higher-quality part-of-speech representations than ELMo2. In particular, word identities are strong features in part-of-speech tagging when used in combination with other indicators; since ELMo1 is closer to the word representations than ELMo2, it may be easier to identify word identities from it, meaning the probe may utilize word identities more readily, as opposed to picking up on a representation of part-of-speech.

3.4.1 Experiments

We run experiments on the first and second contextual layers of ELMo, denoted ELMo1 and ELMo2. We also examine the representations of an untrained BiLSTM run on the non-contextual character CNN word embeddings of ELMo, shown to be a strong baseline contextualization method, but without any linguistic knowledge learned from context (Zhang and Bowman, 2018; Hewitt and Manning, 2019). We denote this model Proj0.

We train linear and MLP-1 probes for part-of-speech tagging, and bilinear and MLP-1 probes for dependency edge prediction, all with default hyperparameters (§ 3.3.2). We examine both the linguistic task accuracy and selectivity achieved by each probe on each representation.

3.4.2 Results & Discussion

We find probes on ELMo2 to be strikingly more selective than those on ELMo1, consistent across all probes, both for part-of-speech tagging and dependency head prediction. In particular, the linear probe on ELMo2 achieves selectivity of 31.4, compared to selectivity of 26.0 for ELMo1, for a gain of 5.4. The same probe achieves 96.6 linguistic task accuracy on ELMo2 and 97.2 on ELMo1, for a loss of 0.6. The MLP probe shows roughly the same result. So, does ELMo1 have a better grasp of part-of-speech than ELMo2? Our results, summarized in Table 3.2, offer the alternative hypothesis that probes use word identity as a feature to predict part-of-speech, and that feature is less easily available in ELMo2 than ELMo1.

Probes on Proj0 and ELMo2 achieve similar part-of-speech tagging accuracy, echoing findings of (Zhang and Bowman, 2018), but we find that Proj0 is far less selective, suggesting that probes on ELMo2 rely far less on word identities than those on Proj0. Without considering selectivity, it might be thought that ELMo2

Part-of-speech Tagging				
	Linear		MLP-1	
Model	Accuracy	Selectivity	Accuracy	Selectivity
Proj0	96.3	20.6	97.1	1.6
ELMo1	97.2	26.0	97.3	4.5
ELMo2	96.6	31.4	97.0	8.8

Dependency Edge Prediction				
	Bilinear		MLP-1	
Model	Accuracy	Selectivity	Accuracy	Selectivity
Proj0	79.9	-4.3	86.5	-9.0
ELMo1	89.7	6.7	92.5	-1.0
ELMo2	84.5	6.2	89.5	1.4

Table 3.2: Part-of-speech and dependency edge prediction probe accuracies and selectivities across three representations. ELMo1 and ELMo2 are the two contextual layers of ELMo, while Proj0 refers to an untrained BiLSTM contextualization of ELMo's non-contextual character CNN representations.

encodes nothing about part-of-speech, since it doesn't beat the Proj0 random representation baseline. Taking selectivity into account, we see that probes on ELMo2 are unable to rely on word identity features like those on Proj0, so to achieve high accuracy, they must rely on emergent properties of the representation.

3.5 Related Work

Early work in probing, (also known as diagnostic classification (Hupkes et al., 2018),) extracted properties like parts-of-speech, gender, tense, and number from distributional word vector spaces like word2vec and GloVe (Mikolov et al., 2013b; Pennington et al., 2014) using linear classifiers (Köhn, 2015; Gupta et al., 2015). Soon after, the investigation of intermediate layers of deep models using linear probes was introduced independently by Ettinger et al. (2016) and Shi et al. (2016) in NLP and Alain and Bengio (2016) in computer vision.

Since then, probing methods have varied as to whether they investigate whole-sentence properties like sentence length and word content using a sentence vector (Shi et al., 2016; Adi et al., 2017; Conneau et al., 2018), word properties like verb tense or part-of-speech using word vectors (Shi et al., 2016; Belinkov et al., 2017; Liu et al., 2019a), or word-pair properties like syntactic relationships using pairs of vectors (Tenney et al., 2019; Hewitt and Manning, 2019). Probes have been used to make relative claims between models or components (Adi et al., 2017; Liu et al., 2019a; Belinkov et al., 2017) or absolute claims about models above baselines. Probes have also been used to test hypotheses about the mechanisms by which models perform tasks (Hupkes et al., 2018; Giulianelli et al., 2018).

Previous work has made extensive use of control representations like non-contextual word embeddings or models with random weights (Belinkov et al., 2017; Tenney et al., 2019; Saphra and Lopez, 2019; Hewitt and Manning, 2019); our control tasks provide a complementary perspective, measuring a probe's ability to decode a random function from the representation of interest.

The most related work to this chapter is that of Zhang and Bowman (2018), who presented experiments for understanding the roles probe training sample size and memorization have on linguistic task accuracy. They observed that untrained BiLSTM contextualizers achieved almost the same part-of-speech tagging accuracies as trained contextualizers, and found that by reducing the probe training set, the trained models could be shown to significantly outperform the untrained model. They evaluated which representations were easiest to memorize from by probing to predict nearby words, finding as we do that word identities are most easily available in untrained contextualizers' representations. They take this as evidence that gains in part-of-speech probing accuracy on the trained representations over the untrained representations are due to linguistic properties, not memorization. Our experiments with selectivity complement their results, finding among other things that even though untrained BiLSTMs are better for memorization than ELMo, there is still a striking capacity for memorization using ELMo when using high-capacity probes.

3.5.1 Random tasks

Zhang et al. (2017) defined completely random tasks related to Rademacher complexity (Bartlett and Mendelson, 2001) to understand the capacity of neural networks to overfit, showing that they are expressive enough to fit random noise, but still function as effective models. In our random control tasks, randomness is applied at the type-level rather than at the example-level, and are designed to have strong non-linguistic structure as opposed to absolutely no structure. While the tasks of Zhang et al. (2017) aid in understanding the expressivity of neural nets, our control tasks aid in understanding the expressivity of a probe model with respect to a specific linguistic task.

3.6 Conclusion

Through probing methods, it has been shown that a broad range of supervised learning tasks can be turned into tools for understanding the properties of contextual word representations (Conneau et al., 2018; Tenney et al., 2019). Alain and Bengio (2016) suggested we may think of probes as "thermometers used to measure the temperature simultaneously at many different locations". We instead emphasize the joint roles of representations and probes together in achieving high accuracy on a task; we suggest that probes be thought of as *craftspeople*; their performance depends not only on the materials they're given, but also on their expressivity.

To explore the relationship between representations, probes, and task accuracies, we defined control tasks, which by construction can only be learned by the probe itself. We've suggested that a probe which provides insights into the properties of the representation should be *selective*, achieving high linguistic task accuracy and low control task accuracy. Selectivity measures the probe's ability to make numerous output decisions independently of linguistic properties of the representation.

We've found that linear and bilinear models achieve higher selectivity at similar accuracy to MLP probes on part-of-speech tagging. MLP probes, achieving higher accuracy on the more complex task of dependency edge prediction, can be re-designed to achieve higher selectivity at a relatively small cost to dependency edge accuracy, but often not through dropout, the most popular MLP probe regularization method.

Finally, we showed how selectivity can be used to provide added context to probing results, demonstrating that marginal differences in part-of-speech tagging accuracy between ELMo1 and ELMo2 correspond to large differences in selectivity, and similarly, the even though ELMo2 achieves similar part-of-speech tagging accuracy to a random representation baseline, ELMo2 achieves it with much higher selectivity.

However, the methods presented in this chapter are best described as proof that there is a problem with complex probes, and a way of showing the extent to which that problem occurs – these methods are not a foundation on which to understand probing more broadly. In the next chapter, we attempt to provide just this foundation, and solve a broader class of problems with probing that includes those presented in this chapter via a single family of methods with a clear interpretation.[6]

Retrospective. In the years since this work was originally published, there has been considerable discussion of the implications for probing. The work of Pimentel et al. (2020b) in particular critiqued the ideas in this chapter. It suggested that (1) the goal of probing is to estimate mutual information (Shannon, 1948), and thus that (2) we should really use as complex a probe as possible, and finally that (3) probing is not a useful scientific inquiry, since the mutual information of a representation of input text cannot *increase* as a function of the model's processing. Our distinctions between a probe learning a task and the representation (easily) encoding it are thus not meaningful. We discuss our alternative framework in Chapter 4, but in summary, the issue is at (1)—the goal of probing is not to measure mutual information. Mutual information between two random variables is invariant to the representations of those variables; as such it's not a likely candidate for a framework for understanding the properties of representations. There are many ways to encode the idea of *easy extraction* of a property from a representation; Voita and Titov (2020) provide a framework based on the ease of learning as a function of the number of samples. Our work in Chapter 4 focuses instead on ease of extraction through simple (and researcher-specified) function families corresponding to hypotheses about how a property is encoded in a representation.

[6]All code, data, and experiments are available at `https://worksheets.codalab.org/worksheets/0xb0c351d6f1ac4c51b54f1023786bf6b2`.

Chapter 4

Conditional Probing: Measuring Usable Information Beyond a Baseline

4.1 Introduction

In Chapter 3, we showed that highly expressive probes could pick up on signals from the input and use them to distinguish between random classes, or properties, that the language model being studied could not have learned. This is a core problem for the *probing* methodology, since it attempts to relate neural representations to well-understood properties.

To recall, probing analyzes a representation by using it as input into a supervised classifier, which is trained to predict a property, such as part-of-speech (Shi et al., 2016; Ettinger et al., 2016; Alain and Bengio, 2016; Adi et al., 2017; Belinkov, 2021). One suggests that a representation encodes a property of interest if probing that representation produces higher accuracy than probing a baseline representation like non-contextual word embeddings. However, consider a representation that encodes *only* the part-of-speech tags that aren't determined by the word identity. Probing would report that this representation encodes *less about part-of-speech* than the non-contextual word baseline, since ambiguity is relatively rare. Yet, this representation clearly encodes interesting aspects of part-of-speech. How can we capture this?

In this chapter, we present a simple probing method to explicitly condition on a baseline.[1] For a representation and a baseline, our method trains two probes: (1) on just the baseline, and (2) on the concatenation of the baseline and the representation. The performance of probe (1) is then subtracted from that of probe (2). We call this process *conditional probing*. Intuitively, the representation is not penalized for *lacking* aspects of the property accessible in the baseline.

We then theoretically ground our probing methodology in $\mathcal{V}$-information, a theory of *usable* information

[1]Our code is available at `https://github.com/john-hewitt/conditional-probing`.

introduced by Xu et al. (2020) that we additionally extend to multiple predictive variables. We use $\mathcal{V}$-information instead of mutual information (Shannon, 1948; Pimentel et al., 2020b) because any injective deterministic transformation of the input has the same mutual information as the input. For example, a representation that maps each unique sentence to a unique integer must have the same mutual information with any property as does BERT's representation of that sentence, yet the latter is more useful. In contrast, $\mathcal{V}$-information is defined with respect to a family of functions $\mathcal{V}$ that map one random variable to (a probability distribution over) another. $\mathcal{V}$-information can be constructed by deterministic transformations that make a property more accessible to the functions in the family. We show that conditional probing provides an estimate of *conditional $\mathcal{V}$-information* $I_{\mathcal{V}}(\text{repr} \rightarrow \text{property} \mid \text{baseline})$.

In a case study, we answer an open question posed in Chapter 3: how are the aspects of linguistic properties that *aren't explainable by the input layer* accessible across the rest of the layers of the network? We find that the part-of-speech information not attributable to the input layer remains accessible much deeper into the layers of ELMo (Peters et al., 2018a) and RoBERTa (Liu et al., 2019b) than the overall property, a fact previously obscured by the gradual loss across layers of the aspects attributable to the input layer. For the other properties, conditioning on the input layer does not change the trends across layers.

4.2 Conditional $\mathcal{V}$-information Probing

In this section, we describe probing methods and introduce conditional probing. We then review $\mathcal{V}$-information and use it to ground probing.

4.2.1 Probing setup

We start with some notation. Let $X \in \mathcal{X}$ be a random variable taking the value of a sequence of tokens. Let $\phi(X)$ be a representation resulting from a deterministic function of X; for example, the representation of a single token from the sequence in a layer of BERT (Devlin et al., 2019). Let $Y \in \mathcal{Y}$ be a property (e.g., part-of-speech of a particular token), and $\mathcal{V}$ a *probe family*, that is, a set of functions $\{f_\theta : \theta \in \mathbb{R}^p\}$, where $f_\theta : z \rightarrow \mathcal{P}(\mathcal{Y})$ maps inputs z to probability distributions over the space of the label.[2] The input $z \in \mathbb{R}^m$ may be in the space of $\phi(X)$, that is, $\mathbb{R}^d$, or another space, e.g., if the probe takes the concatenation of two representations. In each experiment, a training dataset $\mathcal{D}_{\text{tr}} = \{(x_i, y_i)\}_i$ is used to estimate θ, and the probe and representation are evaluated on a separate dataset $\mathcal{D}_{\text{te}} = \{(x_i, y_i)\}_i$. We refer to the result of this evaluation on some representation R as $\text{Perf}(R)$.

4.2.2 Baselined probing

Let $B \in \mathbb{R}^d$ be a random variable representing a baseline (e.g., non-contextual word embedding of a particular token.) A common strategy in probing is to take the difference between a probe performance on the

[2]We discuss mild constraints on the form that $\mathcal{V}$ can take in the Appendix. Common probe families including linear models and feed-forward networks meet the constraints.

representation and on the baseline (Zhang and Bowman, 2018); we call this **baselined probing performance**:

$$\text{Perf}(\phi(X)) - \text{Perf}(B). \tag{4.1}$$

This difference in performances estimates how much more *accessible* Y is in $\phi(X)$ than in the baseline B, under probe family $\mathcal{V}$.

But what if B and $\phi(X)$ capture distinct aspects of Y? For example, consider if $\phi(X)$ captures parts-of-speech that aren't the most common label for a given word identity, while B captures parts-of-speech that are the most common for the word identity. Baselined probing will indicate that $\phi(X)$ explains less about Y than the baseline, a "negative" probing result. But clearly $\phi(X)$ captures an interesting aspect of Y; we aim to design a method that measures just what $\phi(X)$ *contributes beyond* B in predicting Y, not what B has and $\phi(X)$ lacks.

4.2.3 Our proposal: conditional probing

In our proposed method, we again train two probes; each is the concatenation of two representations of size d, so we let $z \in \mathbb{R}^{2d}$. The first probe takes as input $[B; \phi(X)]$, that is, the concatenation of B to the representation $\phi(X)$ that we're studying. The second probe takes as input $[B; \mathbf{0}]$, that is, the concatenation of B to the $\mathbf{0}$ vector. The conditional probing method takes the difference of the two probe performances, which we call **conditional probing performance**:

$$\text{Perf}([B; \phi(X)]) - \text{Perf}([B; \mathbf{0}]). \tag{4.2}$$

Including B in the probe with $\phi(X)$ means that $\phi(X)$ only needs to contribute what is missing from B. In the second probe, the $\mathbf{0}$ is used as a placeholder, representing the lack of knowledge of $\phi(X)$; its performance is subtracted so that $\phi(X)$ isn't given credit for what's explainable by B.[3]

4.2.4 $\mathcal{V}$-information

$\mathcal{V}$-information is a theory of *usable* information—that is, how much knowledge of random variable Y can be extracted from r.v. R when using functions in $\mathcal{V}$, called a *predictive family* (Xu et al., 2020). Intuitively, by explicitly considering computational constraints, $\mathcal{V}$-information can be *constructed* by computation, in particular when said computation makes a variable easier to predict. If $\mathcal{V}$ is the set of all functions from the space of R to the set of probability distributions over the space of Y, then $\mathcal{V}$-information is mutual information (Xu et al., 2020). However, if the predictive family is the set of all functions, then no representation is more useful than another provided they are related by a bijection. By specifying a $\mathcal{V}$, one makes a hypothesis about the functional form of the relationship between the random variables R and Y. One could let $\mathcal{V}$ be, for example, the set of log-linear models.

[3]The value $\mathbf{0}$ is arbitrary; any constant can be used, or one can train the probe on just B.

Using this predictive family $\mathcal{V}$, one can define the uncertainty we have in Y after observing R as the $\mathcal{V}$-entropy:

$$H_{\mathcal{V}}(Y|R) = \inf_{f \in \mathcal{V}} \mathbb{E}\big[-\log f[r](y) \big], \tag{4.3}$$

where $f[r]$ produces a probability distribution over the labels. Information terms like $I_{\mathcal{V}}(R \to Y)$ are defined analogous to Shannon information, that is, $I_{\mathcal{V}}(R \to Y) = H_{\mathcal{V}}(Y) - H_{\mathcal{V}}(Y|R)$. For brevity, we leave a full formal description, as well as our redefinition of $\mathcal{V}$-information to multiple predictive variables, to the appendix.

4.2.5 Probing estimates $\mathcal{V}$-information

With a particular performance metric, baselined probing estimates a difference of $\mathcal{V}$-information quantities. Intuitively, probing specifies a function family $\mathcal{V}$, training data is used to find $f \in V$ that best predicts Y from $\phi(X)$ (the infimum in Equation A.2), and we then evaluate how well Y is predicted. If we use the negative cross-entropy loss as the Perf function, then **baselined probing** estimates

$$I_{\mathcal{V}}(\phi(X) \to Y) - I_{\mathcal{V}}(B \to Y),$$

the difference of two $\mathcal{V}$-information quantities. This theory provides methodological best practices as well: the form of the family $\mathcal{V}$ should be chosen for theory-external reasons,[4] and since the probe training process is approximating the infimum in Equation 4.3, we're not concerned with sample efficiency.

Baselined probing appears in existing information-theoretic probing work: Pimentel et al. (2020b) define conditional mutual information quantities wherein a lossy transformation $c(\cdot)$ is performed on the sentence (like choosing a single word), and an estimate of the gain from knowing the rest of the sentence is provided; $I(\phi(X); Y|c(\phi(X))) = I(X; Y|c(X))$.[5] Methodologically, despite being a conditional information, this is identical to baselined probing, training one probe on just $\phi(X)$ and another on just $c(\phi(X))$.[6]

4.2.6 Estimating conditional information

Inspired by the transparent connections between $\mathcal{V}$-information and probes, we ask what the $\mathcal{V}$-information analogue of conditioning on a variable in a mutual information, that is, $I(X, Y|B)$. To do this, we extend

[4]There are also PAC bounds (Valiant, 1984) on the estimation error for $\mathcal{V}$-information (Xu et al., 2020); simpler families $\mathcal{V}$ with lower Rademacher complexity result in better bounds.

[5]Equality depends on the injectivity of ϕ; otherwise knowing the representation $\phi(X)$ may be strictly less informative than knowing X.

[6]This is because of the data processing inequality and the fact that $c(\phi(X))$ is a deterministic function of $\phi(X)$.

$\mathcal{V}$-information to multiple predictive variables, and design conditional probing (as presented) to estimate

$$\begin{aligned}&I_{\mathcal{V}}(\phi(X) \to Y|B)\\&\quad= H_{\mathcal{V}}(Y|B) - H_{\mathcal{V}}(Y|B, \phi(X)),\end{aligned}$$

thus having the interpretation of probing what $\phi(X)$ explains about Y apart from what's already explained by B (as can be accessed by functions in $\mathcal{V}$). Methodologically, the innovation is in providing B to the probe on $\phi(X)$, so that the information accessible in B need not be accessible in $\phi(X)$.

4.3 Related Work

Probing—mechanically simple, but philosophically hard to interpret (Belinkov, 2021)—has led to a number of information-theoretic interpretations.

Pimentel et al. (2020b) claimed that probing should be seen as estimating mutual information $I(\phi(X); Y)$ between representations and labels. This raises an issue, which Pimentel et al. (2020b) notes: due to the data processing inequality, the MI between the representation of a sentence (from e.g., BERT) and a label is upper-bounded by the MI between the sentence itself and the label. Both an encrypted document X and an unencrypted version $\phi(X)$ provide the same mutual information with the topic of the document Y. This is because MI allows unbounded work in using X to predict Y, including the enormous amount of work (likely) required to decrypt it without the secret key. Intuitively, we understand that $\phi(X)$ is more useful than X, and that this is because the function ϕ performs useful "work" for us. Likewise, BERT can perform useful work to make interesting properties more accessible. While Pimentel et al. (2020b) conclude from the data processing inequality that probing is not meaningful, we conclude that estimating mutual information is not the goal of probing.

Voita and Titov (2020) propose a new probing-like methodology, *minimum description length (MDL) probing*, to measure the number of bits required to transmit both the specification of the probe and the specification of labels. Intuitively, a representation that allows for more efficient communication of labels (and probes used to help perform that communication) has done useful "work" for us. Voita and Titov (2020) found that by using their methods, probing practitioners could pay less attention to the exact functional form of the probe. $\mathcal{V}$-information and MDL probing complement each other; $\mathcal{V}$-information does not measure sample efficiency of learning a mapping from $\phi(X)$ to Y, instead focusing solely on how well any function from a specific family (like linear models) allows one to predict Y from $\phi(X)$. Further, in practice, one must choose a family to optimize over even in MDL probing; the complexity penalty of communicating the member of the family is analogous to choosing $\mathcal{V}$. Further, our contribution of conditional probing is orthogonal to the choice of probing methodology; it could be used with MDL probing as well.

$\mathcal{V}$-information places the functional form of the probe front-and-center as a *hypothesis* about how structure is encoded. This intuition is already popular in probing. For example, Hewitt and Manning (2019) (Chapter 2) proposed that syntax trees may emerge as squared Euclidean distance under a linear transformation. Further work refined this, showing that a better structural hypothesis may be hyperbolic (Chen et al., 2021) axis-aligned after scaling (Limisiewicz and Mareček, 2021), or an attention-inspired kernel space (White et al., 2021).

In this work, we intentionally avoid claims as to the "correct" functional family $\mathcal{V}$ to be used in conditional probing. Some work has argued for simple probe families (Hewitt and Liang, 2019; Alain and Bengio, 2016), others for complex families (Pimentel et al., 2020b; Hou and Sachan, 2021). Pimentel et al. (2020a) argues for choosing multiple points along an axis of expressivity, while Cao et al. (2021) define the family through the weights of the neural network. Other work performs structural analysis of representations without direct supervision (Saphra and Lopez, 2019; Wu et al., 2020).

In Chapter 3, we suggested that differences in ease of identifying the word identity across layers could impede comparisons between the layers; our conditional probing provides a direct solution to this issue by conditioning on the word identity. Kuncoro et al. (2018) and Shapiro et al. (2021) use control tasks, and Rosa et al. (2020) measures word-level memorization in probes. Finally, under the possible goals of probing proposed by Ivanova et al. (2021), we see $\mathcal{V}$-information as most useful in *discovering emergent structure*, that is, parsimonious and surprisingly simple relationships between neural representations and complex properties.

4.4 Experiments

In our experiments, we aim for a case study in understanding how conditioning on the non-contextual embeddings changes trends in the accessibility of linguistic properties across the layers of deep networks.

4.4.1 Tasks, models, and data

Tasks. We train probes to predict five linguistic properties, roughly arranged in order from lower-level, more concrete properties to higher-level, more abstract properties. We predict five linguistic properties Y: (i) **upos**: coarse-grained (17-tag) part-of-speech tags (Nivre et al., 2020), (ii) **xpos**: fine-grained English-specific part-of-speech tags, (iii) **dep rel**: the label on the Universal Dependencies edge that governs the word, (iv) **ner**: named entities, and (v) **sst2**: sentiment.

Data. All of our datasets are composed of English text. For all tasks except sentiment, we use the Ontonotes v5 corpus (Weischedel et al., 2013), recreating the splits used in the CoNLL 2012 shared task, as verified against the split statistics provided by Strubell et al. (2017).[7][8] Since Ontonotes is annotated with constituency parses, not Universal Dependencies, we use the converter provided in CoreNLP (Schuster and Manning, 2016; Manning et al., 2014). For the sentiment annotation, we use the binary GLUE version (Wang et al., 2019) of

[7]In order to provide word vectors for each token in the corpus, we heuristically align the subword tokenizations of RoBERTa with the corpus-specified tokens through character-level alignments, following Tenney et al. (2019).

[8]Ontonotes uses the destructive Penn Treebank tokenization (like replacing brackets { with `-LCB-` (Marcus et al., 1993)). We perform a heuristic de-tokenization process before subword tokenization to recover some naturalness of the text.

	Baselined		Conditional	
	ϕ_1	ϕ_2	ϕ_1	ϕ_2
upos	0.20	0.16	0.22	0.20
xpos	0.20	0.16	0.21	0.20
dep rel	0.99	0.81	1.00	0.87
ner	0.24	0.23	0.25	0.24
sst2	0.18	0.13	0.17	0.13

Table 4.1: Results on ELMo, reported in bits of $\mathcal{V}$-information; higher is better. ϕ_i refers to layer i.

the the Stanford Sentiment Treebank corpus (Socher et al., 2013). All results are reported on the development sets.

Models. We evaluate the popular RoBERTa model (Liu et al., 2019b), as provided by the HuggingFace Transformers package (Wolf et al., 2020), as well as the ELMo model (Peters et al., 2018a), as provided by the AllenNLP package (Gardner et al., 2017). When multiple RoBERTa subwords are aligned to a single corpus token, we average the subword vector representations.

Probe families. For all of our experiments, we choose $\mathcal{V}$ to be the set of affine functions followed by softmax.[9] For word-level tasks, we have

$$f_\theta(\phi_i(X)_j) = \text{softmax}(W\phi_i(X)_j + b) \tag{4.4}$$

where i indexes the layer in the network and j indexes the word in the sentence. For the sentence-level sentiment task, we average over the word-level representations, as

$$f_\theta(\phi_i(X)) = \text{softmax}(W\ \text{avg}(\phi_i(X)) + b) \tag{4.5}$$

4.4.2 Results

Results on ELMo. ELMo has a non-contextual embedding layer ϕ_0, and two contextual layers ϕ_1 and ϕ_2, the output of each of two bidirectional LSTMs (Hochreiter and Schmidhuber, 1997). Previous work has found that ϕ_1 contains more syntactic information than ϕ_2 (Peters et al., 2018b; Zhang and Bowman, 2018). Baselined probing performance, in Table 4.1, replicates this finding. But in Chapter 3, we conjecture that this may be due to accessibility of information from ϕ_0. Conditional probing answers shows that when only measuring information not available in ϕ_0, there is still more syntactic information in ϕ_1 than ϕ_2, but the difference is much smaller.

[9]We used the Adam optimizer (Kingma and Ba, 2014) with starting learning rate 0.001, and multiply the learning rate by 0.5 after each epoch wherein a new lowest validation loss is not achieved.

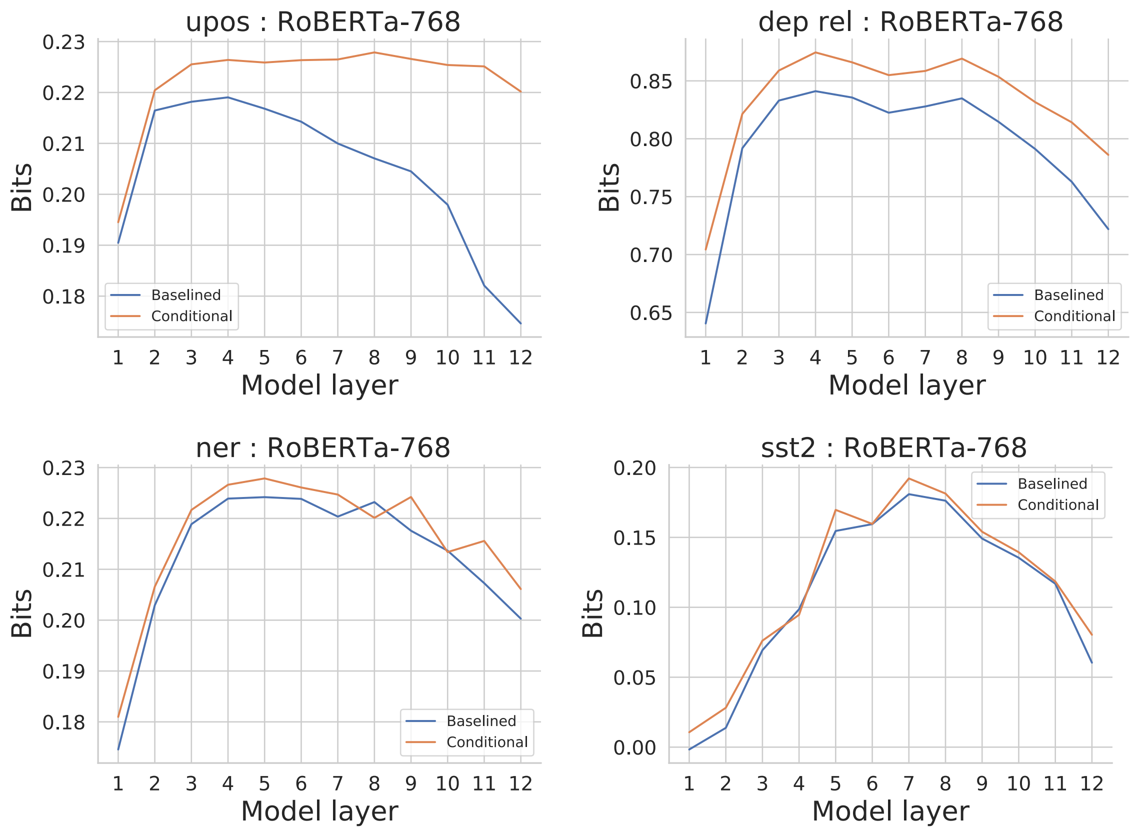


Figure 4.1: Probing results on RoBERTa. Results are reported in bits of $\mathcal{V}$-information; higher is better.

Results on RoBERTa. RoBERTa-base is a pretrained Transformer consisting of a word-level embedding layer ϕ_0 and twelve contextual layers ϕ_i, each the output of a Transformer encoder block (Vaswani et al., 2017). We compare baselined probing performance to conditional probing performance for each layer. In Figure 4.1, baselined probing indicates that part-of-speech information decays in later layers. However, conditional probing shows that information *not* available in ϕ_0 is maintained into deeper layers in RoBERTa, and only the information already available in ϕ_0 decays. In contrast for dependency labels, we find that the difference between layers is lessened after conditioning on ϕ_0, and for NER and sentiment, conditioning on ϕ_0 does not change the results.

4.5 Conclusion

In this chapter, we proposed *conditional probing*, a simple method for conditioning on baselines in probing studies, and grounded the method theoretically in $\mathcal{V}$-information. In a case study, we found that after conditioning on the input layer, usable part-of-speech information remains much deeper into the layers of ELMo and RoBERTa than previously thought, answering an open question from Chapter 3 (Hewitt and Liang, 2019). Conditional probing is a tool that practitioners can easily use to gain additional insight into representations.[10]

Whereas in Chapter 2 we presented a simple baselined probe and interesting result on neural language models learning syntax, and in Chapter 3 we presented problems with complex probes and a way of measuring those problems, in this chapter we finally provided a foundation for future probing studies, and a meaningful answer to what probing is doing: estimating usable information. By designing baselines around whatever property one does *not* want to study in probing, one can use conditional probes to estimate conditional usable information and come to reliable conclusions. This ends the portion of this thesis centered around probing; the method is certainly useful, but can only tell is where in a network a property is easy to decode. In the next two chapters, we'll take a more active approach to understanding neural networks by designing and evaluating ways of making surgical changes to them.

[10]An executable version of the experiments in this chapter is on CodaLab, at this link: `https://worksheets.codalab.org/worksheets/0x46190ef741004a43a2676a3b46ea0c76`.

Part II

Understanding by Design

Chapter 5

Backpack Language Models

5.1 Introduction

In the first part of this thesis, we presented methods for the discovery of structure in language models. This and similar lines of work treat models as provided to us, such that we must attempt to understand them as they are. This is often the case, but not always. By designing new models with the goal of understanding from the outset, we can as engineers attempt to make the latter goal of understanding the resulting language models easier. This, however, must not come at too much of a cost of the models' overall capabilities or cost. In this chapter, we present a neural network architecture that designs new understanding tools while maintaining most of the power of an existing popular architecture, the Transformer.

Consider the prefix *The CEO believes that* ___, and the problem of debiasing a neural language model's distribution over *he/she*. Intuitively, the bias for *he* originates in the word *CEO*, because replacing *CEO* with *nurse* flips the observed bias. A successful intervention to debias *CEO* must reliably apply in all contexts in which the word *CEO* appears; ideally we would want to make a **non-contextual** change to the model that has predictable effects in **all contexts**. In general, in all aspects of interpretability and control, it is desirable to make interventions with a tractable interface (e.g., non-contextual representations) that apply globally.

Such interventions are difficult in Transformer models (Vaswani et al., 2017) because their contextual representations are monolithic functions of their input. Almost any intervention on the model has complex, non-linear effects that depend on context. We would instead like models that enable precise, rich interventions that apply predictably in all contexts, and are still expressive, so they are a viable alternative to Transformers.

We address these challenges with a new neural architecture, the *Backpack*, for which predictions are log-linear combinations of non-contextual representations. We represent each word in a vocabulary as a set of non-contextual *sense vectors* that represent distinct learned aspects of the word. For example, sense vectors for the word "science" could encode types of science, connections to technology, notions of science being "settled," or different aspects of the scientific process (replication or experiment) (Table 5.1). Sense vectors do not learn classic word sense, but more general aspects of a word's potential roles in different contexts; in fact,

Figure 5.1: Transformers are monolithic functions of sequences. In Backpacks, the output is a weighted sum of non-contextual, learned word aspects.

A few senses of the word *science*				
Sense 3	Sense 7	Sense 9	Sense 10	Sense 8
fiction	replication	religion	settled	clones
fictional	citation	rology	sett	experiments
Fiction	Hubble	hydra	settle	mage
literacy	reprodu	religions	unsett	experiment
denial	Discovery	nec	Sett	rats

$\textbf{\textit{MacBook}}_{\textbf{\textit{HP}}}$ = ***MacBook*** $-$ ***Apple*** $+$ ***HP***
The MacBook is best known for its form factor, but HP has continued with its Linux-based computing strategy. HP introduced the Hyper 212 in 2014 and has continued to push soon-to-be-released 32-inch machines with Intel's Skylake processors.

Table 5.1: Examples of the rich specialization of sense vectors representing the word *science*, and an example of editing sense vectors non-contextually (changing MacBook to be associated with HP) and having the resulting *contextual* predictions change.

they can be seen as a multi-vector generalization of classic word vectors (Mikolov et al., 2013a).[1]

To make interventions on sense vectors behave predictably in different contexts, a Backpack represents each word in a sequence as a **linear combination** of the sense vectors for all words in the sequence. The expressivity of a Backpack comes from the network that computes the weights of the linear combination as a function of the whole sequence; for example, in all our experiments we use a Transformer for this. Since sense vectors are softly selected depending on the context, they can specialize; each sense can learn to be predictively useful in only some contexts. The log-linear contribution of senses to predictions then implies that the interventions on sense vectors we demonstrate in Section 5.6 apply identically (up to a non-negative scalar weight) regardless of context.

Our experiments demonstrate the expressivity of Backpack language models, and the promise of interventions on sense vectors for interpretability and control. In Section 5.4 we train Backpack language models on 50B tokens (5 epochs) of OpenWebText; a Backpack with 124M parameters in the contextual network (and 46M parameters for sense vectors) achieves the perplexity of a 124M-parameter Transformer; thus one pays for more interpretability with a larger model size. In Section 5.5, we show that sense vectors specialize to encode rich notions of word meaning. Quantitatively, on four lexical similarity datasets (e.g., SimLex999), sense vectors of a 170M parameter Backpack outperform word embeddings of the 6B-parameter GPT-J-6B Transformer, and approach the performance of state-of-the-art specialized methods for this task. Finally, in Section 5.6 we show that sense vectors offer a control mechanism for Backpack language models. For example, stereotypically gendered profession words (e.g., “CEO” or “nurse”) tend to learn a sense vector associated with this gender bias; by downscaling this sense vector, we greatly reduce disparity in contextual predictions in a limited setting.

5.2 The Backpack Architecture

In this section, we define the general form of the Backpack architecture. We then show how continuous bag-of-words word2vec (CBOW) (Mikolov et al., 2013a) and Self-Attention-Only networks (Elhage et al., 2021; Olsson et al., 2022) are special cases of Backpacks.

5.2.1 Backpack General Form

A Backpack is a parametric function that maps a sequence of symbols $\mathbf{x}_{1:n} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ to a sequence of vectors $\mathbf{o}_{1:n} = (\mathbf{o}_1, \ldots, \mathbf{o}_n)$, where each symbol $\mathbf{x}_i$ belongs to a finite vocabulary $\mathcal{V}$ and $\mathbf{o}_i \in \mathbb{R}^d$. We call $\mathbf{o}_i$ the *Backpack representation* of $\mathbf{x}_i$ in the context of a sequence $\mathbf{x}_{1:n}$.

Sense vectors. For each $\mathbf{x} \in \mathcal{V}$, a Backpack constructs k *sense* vectors

$$C(\mathbf{x})_1, \ldots, C(\mathbf{x})_k, \tag{5.1}$$

[1]Our code, sense vectors, language model weights, and demos are available at `https://backpackmodels.science`.

where $C : \mathcal{V} \to \mathbb{R}^{k \times d}$. Sense vectors are a multi-vector analog to classic non-contextual word representations like word2vec or GloVe: we make this analogy precise in Section 5.2.2.

Weighted sum. For a sequence $\mathbf{x}_{1:n}$, the representation $\mathbf{o}_i$ of element $\mathbf{x}_i$ is a weighted sum of the predictive sense vectors for the words in its context: given *contextualization weights* $\alpha \in \mathbb{R}^{k \times n \times n}$,

$$\mathbf{o}_i = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell i j} C(\mathbf{x}_j)_\ell. \tag{5.2}$$

The contextualization weights $\alpha_{\ell i j}$ of a Backpack are themselves defined by a (non-linear) *contextualization function* of the entire sequence $\mathbf{x}_{1:n}$:

$$\alpha = A(\mathbf{x}_{1:n}), \tag{5.3}$$

where $A : \mathcal{V}^n \to \mathbb{R}^{k \times n \times n}$.

The name "Backpack" is inspired by the fact that a backpack is like a bag—but more orderly. Like a bag-of-words, a Backpack representation is a sum of non-contextual senses; but a Backpack is more orderly, because the weights in this sum depend on the ordered sequence.

Backpack Models. A *Backpack model* is a probabilistic model that defines probabilities over some output space $\mathcal{Y}$ as a log-linear function of a Backpack representation $\mathbf{o}_{1:n} \in \mathbb{R}^{n \times d}$:

$$p(\mathbf{y}|\mathbf{o}_{1:n}) = \text{softmax}\left(E(\mathbf{o}_{1:n})\right), \tag{5.4}$$

where $\mathbf{y} \in \mathcal{Y}$ and $E : \mathbb{R}^{n \times d} \to \mathbb{R}^{|\mathcal{Y}|}$ is a linear transformation. Because Backpack models are log-linear in their representations, the sense vectors contribute log-linearly to predictions. This allows us to inspect a sense vector by projecting it onto the vocabulary via E and observe exactly how it will contribute to predictions in any context.

Models parameterized by the prevailing deep neural architectures—including LSTMs (Hochreiter and Schmidhuber, 1997) and Transformers—are not Backpacks because their output representations are (relatively) unconstrained functions of the entire sequence. By contrast, Backpack models may seem limited in expressivity: the representations $\mathbf{o}_i$ are scalar-weighted sums of non-contextual vectors $C(\mathbf{x}_j)_\ell$. Contextual relationships between sequence elements can only be expressed through the weights $\alpha = A(\mathbf{x}_{1:n})$. Nevertheless, our experiments show that an expressive contextualization weight network can represent complex functions by weighted sums of sense vectors, e.g., our 170M parameter Backpack LM uses a 124M-parameter Transformer to compute α, and achieves the loss of a 124M-parameter Transformer LM.

To place Backpacks in some historical context, we now show how two existing architectures can be described as Backpacks.

5.2.2 Continuous Bag-of-Words is a Backpack

The continuous bag-of-words word2vec model defines a probability distribution over a center word $\mathbf{x}_c \in \mathcal{V}$ conditioned on n context words $\mathbf{x}_{1:n}$.[2] The model proceeds to (1) construct vector embeddings $\mathbf{v}_{\mathbf{x}}$ for each $\mathbf{x} \in \mathcal{V}$, and (2) uniformly average the embeddings of the context words to predict the center word:

$$\overline{\mathbf{v}}_{\mathbf{x}_c} = \sum_{i=1}^{n} \frac{1}{n} \mathbf{v}_{\mathbf{x}_i}, \tag{5.5}$$

$$p(\mathbf{x}_c \mid \mathbf{x}_{1:n}) = \text{softmax}(U \overline{\mathbf{v}}_{\mathbf{x}_c}), \tag{5.6}$$

where $U \in \mathbb{R}^{\mathcal{V} \times d}$. We see that $\overline{\mathbf{v}}_{\mathbf{x}_c}$ is a Backpack representation by setting $C(\mathbf{x}) = \mathbf{v}_{\mathbf{x}} \in \mathbb{R}^{1 \times d}$ in Equation (5.1) using a single sense vector ($k = 1$) and setting the contextualization weights in Equation (5.3) to be uniform: $\alpha_{\ell ij} = \frac{1}{n}$.

This connection to CBoW foreshadows the emergence of linguistic structures in the predictive sense vectors of Backpack models, just as these structures emerge in CBoW (Mikolov et al., 2013a), as we discussed in Chapter 1.

5.2.3 Single-Layer Self-Attention is a Backpack

The Backpack structure—define sense vectors (values), and use the sequence to determine how to sum them (weights)—may remind the reader of a single layer of self-attention. The key-query-value self-attention function is as follows:

$$\mathbf{o}_j = \sum_{i=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell ij} O V^{(\ell)} \mathbf{x}_j \tag{5.7}$$

$$\alpha_\ell = \text{softmax}(\mathbf{x}^\top K^{(\ell)\top} Q^{(\ell)} \mathbf{x}), \tag{5.8}$$

where $\mathbf{x} \in \mathbb{R}^{n \times d}$ is (overloaded) to be a non-contextual embedding of the sequence, $O \in \mathbb{R}^{d \times d/k}$, and $V^{(\ell)} \in \mathbb{R}^{d/k \times d}$, where k is the number of attention heads. The self-attention function is a Backpack with $C(\mathbf{x}_j)_\ell = O V^{(\ell)} \mathbf{x}_j$. Self-attention-only networks are studied in the context of, e.g., mechanistic interpretability (Elhage et al., 2021). A Transformer composes blocks of self-attention and non-linear feed-forward layers that combine information from the whole sequence; unlike a Transformer, the contextualization weights of a Backpack each select a non-contextual sense of a single word.

5.3 Language Modeling with Backpacks

In this section, we define a neural autoregressive language model parameterized by a Backpack. We use the standard softmax parameterization of the probability over the next token in a sequence, with a weight matrix

[2]Context in this setting is usually defined as words surrounding the center word.

$E \in \mathbb{R}^{d\times|\mathcal{V}|}$ that maps a representation $\mathbf{o}_j \in \mathbb{R}^d$ to logits $E^\top \mathbf{o}_j \in \mathbb{R}^{|\mathcal{V}|}$:

$$p(\mathbf{x}_j \mid \mathbf{x}_{1:j-1}) = \text{softmax}(E^\top \mathbf{o}_j). \tag{5.9}$$

Recall (Section 5.2.1) that Backpack representations $\mathbf{o}_j$ are defined by sense vectors $C(\mathbf{x})$ and contextualization weights α_j. In Section 5.3.1 we describe a parameterization of C for the predictive sense vectors in Equation (5.1), and in Section 5.3.2 we describe a parameterization of A for the contextualization weight network in Equation (5.3). When $\mathbf{o}_j$ is parameterized by a Backpack, we call a model of the form given by Equation (5.9) a *Backpack LM*.

5.3.1 Parameterizing senses

For the sense function $C : \mathcal{V} \to \mathbb{R}^{k\times d}$, we embed each $\mathbf{x} \in \mathcal{V}$ into $\mathbb{R}^d$ and pass these embeddings though a feed-forward network FF : $\mathbb{R}^d \to \mathbb{R}^{k\times d}$:

$$C(\mathbf{x}) = \text{FF}(E\mathbf{x}), \tag{5.10}$$

where the embedding/projection matrix E is tied to the output matrix in Equation (5.9) (Press and Wolf, 2017). Note that we could define all $k \times |\mathcal{V}|$ sense vectors using a lookup table, but this would be an enormous number of parameters as k grows large. Instead, we embed the words as $E\mathbf{x} \in \mathbb{R}^d$, and then blow them up to $\mathbb{R}^{d\times k}$ using shared weights. This may explain the related sense roles observed for different word types in Section 5.5.1.

5.3.2 Parameterizing contextualization weights

We parameterize $A : \mathcal{V}^n \to \mathbb{R}^{k\times n\times n}$ using a standard Transformer, followed by a layer of multi-headed key-query self-attention. That is, we pass an embedded sequence through a Transformer

$$\mathbf{h}_{1:n} = \text{Transformer}(E\mathbf{x}_{1:n}) \tag{5.11}$$

(with proper autoregressive masking and some position representation) and compute $A(\mathbf{x}_{1:n}) = \alpha$, where

$$\alpha_\ell = \text{softmax}(\mathbf{h}_{1:n} K^{(\ell)\top} Q^{(\ell)} \mathbf{h}_{1:n}^\top), \tag{5.12}$$

for each predictive sense $\ell = 1, \ldots, k$ with matrices $K^{(\ell)}, Q^{(\ell)} \in \mathbb{R}^{d\times d/k}$. We can think of the k senses as heads and, for each head, the contextualization weights define a distribution of attention over words.[3]

[3]Note that the sense weights are normalized (1) independently for each sense, and (2) to sum to one over the sequence length.

Model	OpenWebText PPL ↓	LAMBADA PPL ↓	LAMBADA ACC ↑	Wikitext PPL ↓	BLiMP ↑
Backpack-Micro	**31.5**	**110**	**24.7**	**71.5**	75.6
Transformer-Micro	34.4	201	21.3	79.5	**77.8**
Backpack-Mini	**23.5**	**42.7**	**31.6**	**49.0**	76.2
Transformer-Mini	24.5	58.8	29.7	52.8	**80.4**
Backpack-Small	**20.1**	**26.5**	**37.5**	**40.9**	76.3
Transformer-Small	**20.2**	32.7	34.9	42.2	**81.9**

Table 5.2: Language modeling performance; all models trained for 100k steps, 500K token batch size, on OWT. For PPL, lower is better; for accuracy, higher is better. Note that models are not parameter-comparable; each Backpack has a matched-size Transformer in its contextualization network.

5.4 Experiments Training Backpack LMs

In this section we specify the hyperparameters used to train Backpack and Transformer language models (Section 5.4.1), data and optimization procedure (Section 5.4.2), evaluations (Section 5.4.3) and results (Section 5.4.4). We also show the necessity of learning $k > 1$ sense vectors to achieve strong language modeling performance (Section 5.4.5).

5.4.1 Models

We train three Transformer baseline models, which we label Micro (30M parameters), Mini (70M parameters), and Small (124M parameters; the same size as GPT-2 small). We also train Micro (40M), Mini (100M), and Small (170M) Backpack language models, for which the weighting function (Equation 5.11) is parameterized using the corresponding Transformer, and almost all extra parameters are in the non-contextual sense vectors.[4] Backpacks thus cost extra parameters and compute beyond their underlying contextualization network. Except where stated, we use $k = 16$ sense vectors in all Backpacks (Section B.1).

We use a reduced sequence length of 512 for all models, and the 50,257-subword GPT-2 tokenizer. Model hidden dimensionalities, layer counts, and head counts are reported in Table B.3.

5.4.2 Data & Optimization

We train all models on OpenWebText (Gokaslan et al., 2019), a publicly available approximate reconstruction of the English WebText corpus used to train the GPT-2 family of models (Radford et al., 2019). We use a batch size of 524,288 tokens, and train all models for 100,000 gradient steps for a total of 52B tokens; training for longer is known to make marginal difference for small models (Hoffmann et al., 2022). The size of OpenWebText means this is roughly 5 epochs. We use cross-entropy loss and the AdamW optimizer, with a warmup of 5,000 steps and linear decay to zero.

[4]There are a negligible number of additional parameters in the final key-query Backpack operation (Equation 5.12)).

5.4.3 Evaluations

Before our experiments in interpretability and control, we check the expressivity of Backpacks. We evaluate models on perplexity for a held out set of OpenWebText, perplexity and accuracy for the (OpenAI variant of) LAMBADA evaluation of long-distance dependencies (Radford et al., 2019; Paperno et al., 2016), perplexity on Wikitext (Merity et al., 2017), and BLiMP English linguistic competence accuracy (Warstadt et al., 2020) evaluated using the EleutherAI harness (Gao et al., 2021) (Version 1).

5.4.4 Discussion

Comparing each Backpack LM to a Transformer LM of equivalent specification to the Backpack's contextualization network, we see that the Backpack performs roughly as well (Table 5.2). Again, the Backpack has more parameters, a tax for the interface provided by sense vectors. During training, we find that Backpack language models take longer to converge than Transformers. Curiously, while the Small Backpack and Transformer achieve almost identical OWT perplexity, the Backpack language models perform substantially better on LAMBADA and Wikitext, but worse on BLiMP.

5.4.5 Effect of varying the number of senses

To study the impact of the number of sense vectors on language modeling performance, we train Mini-sized Backpack language models on a reduced schedule of 50,000 gradient steps, for $k \in \{1, 4, 16, 64\}$ sense vectors. The perplexities for $k = 1, 4, 16, 64$ are 38.6, 29.3, 26.0, and 24.1, demonstrating the necessity of a non-singleton set of sense vectors. Table B.2 contains the full results.

5.5 Emergent Structure in Sense Vectors

Backpack language model sense vectors are not trained using a supervised notion of word sense, but implicitly specialize to encode different shades of a word's predictive use. In this section, we qualitatively examine sense vectors (Section 5.5.1) and quantitatively demonstrate their effectiveness in computing lexical similarity and relatedness (Section 5.5.2). Taken together, this suggests that sense vectors can provide a high-level interface for intervention, which we explore in Section 5.6.

5.5.1 Visualizing Senses

Empirically, trained Backpack models associate specific sense vector indices with different roles for prediction. We interpret these roles by picking a sense ℓ of a word $\mathbf{x}$, and projecting this sense onto the word embeddings: $E^\top C(\mathbf{x})_\ell \in \mathbb{R}^{|\mathcal{V}|}$. Note that this is *exactly* (up to a scalar) how this sense contributes to any prediction of the model. We interpret a sense vector's role by reporting the words with the highest score under this projection.

Sense 12 (*relatedness*)

tasty	*quickly*	*Apple*	*believe*
tasty	quick	Apple	belief
culinary	quickest	Apple	Belief
tasted	quick	iPhone	beliefs
delicious	quicker	iPhone	believing
taste	fast	iPhones	believe

Sense 14 (*Verb objects, nmod nouns*)

build	*attest*	*importance*	*appreciate*
bridges	worthiness	maintaining	finer
wall	Published	wellbeing	nuance
lasting	superiority	teamwork	beauty
ig	accuracy	plurality	irony
rapport	validity	upholding	simplicity

Sense 3 (*next wordpiece*)

pizza	*interest*	*the*
cutter	rate	slightest
tracker	rates	same
iol	groups	entirety
makers	waivers	rest
maker	waiver	latter

Sense 7 (*Proper Noun Associations*)

Apple	*Obama*	*Messi*
macOS	Dreams	Messi
iCloud	Barack	Argentina
Siri	Ob	Mess
iOS	Michelle	Barcelona
tv	Jeremiah	iesta

Table 5.3: Visualization of how the same sense index across many words encodes fine-grained notions of meaning, relatedness, and predictive utility. Each sense is given a label thought up by the authors, and for a few words, the target words that are highest scored by the sense vector.

Model	SL999	SV3500	RG65	WS353
Classic Non-Contextual Embeddings				
word2vec	0.442	0.367	0.679	0.684
GloVe	0.371	0.227	0.687	0.607
Embeddings from large existing models				
GPT2-1.5B	0.523	0.418	0.670	0.706
GPT-J-6B	0.492	0.374	**0.766**	0.673
Embeddings from our models + baseline Transformer				
Trnsf 124M	0.478	0.363	0.634	0.681
Sim_{12} (ours)	0.522	0.471	0.754	**0.749**
Sim_{14} (ours)	0.500	**0.502**	0.591	0.655
Sim_{min} (ours)	**0.540**	0.471	0.653	0.607
Special-purpose SOTA models				
SOTA (Single)	0.554	0.473	0.835	0.764
SOTA (Multi)	0.605	0.528	-	0.807

Table 5.4: Results on lexical similarity evaluation. All numbers are Spearman correlations; higher is better.

Table 5.3 visualizes a few of these senses. For example, sense 12 seems to encode a broad notion of relatedness for almost all words; sense 3 encodes particulars of the bigram distribution given $\mathbf{x}$; sense 14 seems to encode both associated objects for verbs, and noun modifier dependency children for nouns. In Section 5.5.2 we show that sense 14 encodes a powerful notion of verb similarity.

5.5.2 Lexical Relationship Tests

Classic lexical-relatedness and similarity tests measure the extent to which a similarity function on pairs of words correlates with human-elicitied notions of similarity. Similarity functions derived from word embeddings are evaluated by Spearman correlation between the predicted and true similarity rank-order. Early non-contextual embeddings like COALS (Rohde et al., 2005), word2vec (Mikolov et al., 2013a), and GloVe (Pennington et al., 2014) have recently been outperformed by word embeddings derived by distillation of contextual networks (Bommasani et al., 2020; Gupta and Jaggi, 2021; Chronis and Erk, 2020). We evaluate Backpack LM sense vectors on similarity datasets SimLex999 (Hill et al., 2015), SimVerb3500 (Gerz et al., 2016), and relatedness datasets RG65 (Rubenstein and Goodenough, 1965) and (Agirre et al., 2009).

Sense$_\ell$ Cosine. For all $\ell \in \{1, \dots, k\}$, we define a similarity function based only on sense ℓ:

$$\text{Sim}_\ell(\mathbf{x}, \mathbf{x}') = \text{cossim}(C(\mathbf{x})_\ell, C(\mathbf{x}')_\ell), \tag{5.13}$$

where cossim is cosine similarity. Intuitively, we expect that some senses may specialize to learn lexical relatedness or similarity.

Minimum Sense Cosine. Because each sense encodes a different aspect of a word's meaning, we might expect that highly similar words are similar across *all* senses. We test for this strong form of similarity using

$$\text{Sim}_{\text{min}}(\mathbf{x}, \mathbf{x}') = \min_\ell \text{Sim}_\ell(\mathbf{x}, \mathbf{x}') \tag{5.14}$$

Other methods. We evaluate embeddings from the tied softmax/embedding matrices of the much larger GPT-2-1.5B (Radford et al., 2019) and GPT-J-6B (Wang and Komatsuzaki, 2021), classic word embeddings (from Bommasani et al. (2020)) and state-of-the art specialized methods using either a single vector per word (Gupta and Jaggi, 2021) or many vectors (Chronis and Erk, 2020).

Discussion. Sense 12 (the "synonym" sense) performs well across datasets, matching or outperforming embeddings like GPT-2-1.5B and GPT-J-6B (Except GPT-J-6B on RG-65). Sense 14, the "verb objects" sense, performs best on just verb similarity (VerbSim3500), and the minimum similarity over senses works especially well on noun lexical similarity (SimLex999.) Our methods approach the performance of state-of-the-art methods; despite being trained for a very different task, sense vectors encode substantial lexical information (Table 5.4).

5.6 Sense Vectors for Control

In this section, we demonstrate several proof-of-concept methods that leverage sense vectors for controlling LM behavior.

5.6.1 Topic-controlled generation

Given a bag-of-words target $b \in \mathbb{R}^{|\mathcal{V}|}$, e.g., *arts, culture*, we would like to bias generation towards sequences related to concepts related to these terms. Our algorithm proceeds in three parts. First, we sort sense vectors by log-probability assigned to b, that is, $b^\top(E^\top C(\mathbf{x})_\ell)$.[5] Second, based on the scores, we assign a re-weighting factor δ to each sense; senses with the higher scores weighted more. (See Section B.4 for details.) Third, we generate from the Backpack using the re-weighted sense vectors, reducing δ back to 1 as the topic is introduced. The updated backpack equation is

$$\mathbf{o}_i = \sum_{j=1}^{n}\sum_{\ell=1}^{k} \alpha_{\ell ij}\delta_{\ell ij} C(\mathbf{x}_j)_\ell, \tag{5.15}$$

where $\delta_{ij\ell}$ is the re-weighting. Intuitively, the semantic coherence of sense vectors may imply that upweighting senses with affinity to the target bag-of-words richly upweights related words and topics. We give details as to how we perform the sense re-weighting and the annealing in Section B.4.

Evaluation. We use the label descriptors of the topic classifier of Antypas et al. (2022), with 17 categories (*sports, arts & culture, health,...*), as the bag-of-words for control. We evaluate control accuracy as the percent of generations to which the classifier assigns the correct topic label, and overall generation quality and diversity using MAUVE scores (Pillutla et al., 2021).[6]

Results. We compare to Plug-and-Play Language Models (PPLM; Dathathri et al. (2019)), a considerably slower, gradient-based control method using our Small Transformer model. We generate 500 samples from each model for each topic across a range of strengths of control. We find that sense controlled generation provides at least as strong control as PPLM (Figure 5.2), though the MAUVE scores of the unmodified Transformer are higher than the Backpack.) Results and examples are provided in the Appendix in Tables B.6, B.10, B.11, B.12.

5.6.2 Mitigating gender bias

Through inspection, we learned that sense vector 10 of many stereotypically gendered profession nouns (nurse, CEO, teacher) coherently express the stereotype through pronouns. Table B.7 gives examples of these senses. We attempt to mitigate gender bias in Backpack behavior on these gendered profession nouns by *turning down* sense 10 (multiplying by a scalar less than 1).

[5]We divide this term by the maximum absolute log-probability of the sense vector, $\max_{x\in\mathcal{V}} \mathbf{x}^\top(E^\top C(\mathbf{x})_\ell)$.

[6]We concatenate generations across the 17 categories and compute MAUVE against OpenWebText validation examples.

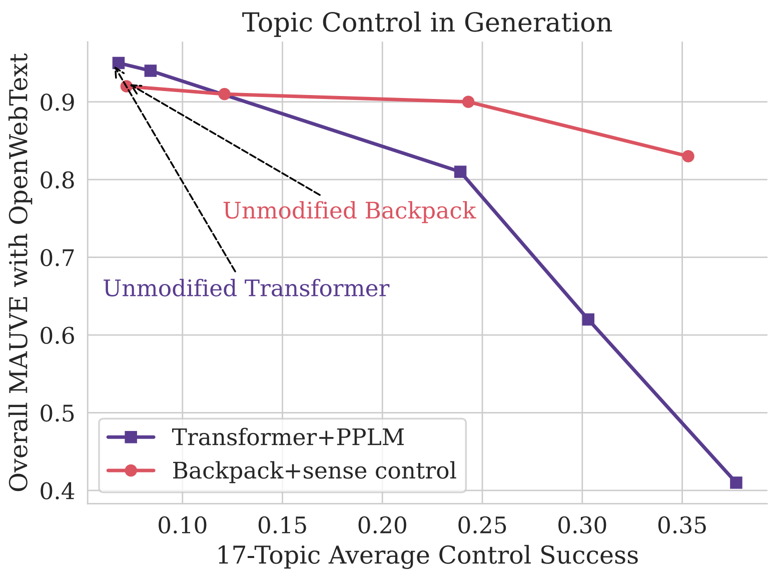


Figure 5.2: Results in controlling topic via sense intervention in Backpacks, and PPLM in Transformers.

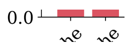


Figure 5.3: The effect on the conditional probability distribution of a Backpack LM on the prefix *when the nurse walked into the room,* of modulating the effect of sense 10 of *nurse* from 0 (totally removed) to 1 (original.)

Model	Bias Ratio ↓	Reduction %
Unbiased	1	-
Transformer		
Unmodified	7.02	-
Project-Nullspace	6.72	5%
Optimize-Nullspace	7.02	0%
Backpack		
Unmodified	4.34	-
Remove-Sense10	2.88	44%
Optimize-Sense10	2.16	65%

Table 5.5: Pronoun-based gender bias reduction in a limited setting.

The MacBook is best known for its form factor, but HP has continued with its Linux-based computing strategy. HP introduced the Hyper 212 in 2014 and has continued to push soon-to-be-released 32-inch machines with Intel's Skylake processors.
The MacBook didn't come into the picture until 2000, when HP followed up with a 15-year flood of HP available laptops.
I was thinking about Brady's role on the Colts before joining other high-profile signings. This is what McElhaney and I discussed. McElhaney: Look, what I didn't mean by this is we didn't move. We think that we're getting a lot better, too.

Table 5.6: Samples from a Backpack wherein *Apple* has been projected out of the *MacBook* sense embeddings, and replaced with *HP*. Likewise with *Brady*, *Patriots*, and *Colts*. Prompts are bolded.

We took an existing set of stereotypically gendered profession nouns from WinoBias (Zhao et al., 2018), and constructed a simplified setting in which a single profession word is in each context, and a third-person nominative pronoun (e.g., he/she/they) is acceptable, e.g., *My CEO said that__*. The full set of nouns and prompts is in Section B.4.2. We evaluate models on the average of the bias of probabilities of *him* vs *her* as follows:

$$\mathop{\mathbb{E}}_{\mathbf{x}\in\text{prompts}}\left[\max\left(\frac{p(\text{he}\mid\mathbf{x})}{p(\text{she}\mid\mathbf{x})},\frac{p(\text{she}\mid\mathbf{x})}{p(\text{he}\mid\mathbf{x})}\right)\right].$$

Baseline. To debias a Transformer with an analogous method, we take inspiration from Bolukbasi et al. (2016). We take $E\mathbf{x}_{\text{he}} - E\mathbf{x}_{\text{she}}$ as an estimate of a gender bias direction, and project the embedding $E\mathbf{x}_{\text{nurse}}$ either to the nullspace of this direction or only partially remove it.

Results. A perfectly unbiased model would achieve ratio 1, whereas the unmodified Transformer achieves 7, and with nullspace projection, 6.72 (Table 5.5). Finding the optimal fraction of the gender bias direction to remove per profession does not improve further. For Backpacks, we find that removing sense 10 from the profession word (setting it to zero) reduces the bias score from 4.34 to 2.88. Learning the optimal removal fraction per profession achieves 2.16, for a total reduction of 65%.[7] In Figure 5.3, we demonstrate the clear effect of ablating sense 10 on the most likely words in one of these contexts.[8]

[7]Curiously, Backpacks are overall less biased to begin with (in this setting); we don't have a strong hypothesis as to why.

[8]It is incidental that sense 10 encodes gender bias as opposed to another sense index; the consistency in index across words may be due to parameter sharing in C.

5.6.3 Knowledge editing

Sense vectors show promise for use in *knowledge editing* (De Cao et al., 2021)—editing a model's predictions about world knowledge. In particular, many associations with proper nouns can be localized to sense vectors in that noun. In this qualitiative proof-of-concept, we edit the sense vectors of a target word $\mathbf{x}$ (e.g., *MacBook* to remove associations with a word $\mathbf{x}_r$ (e.g., *Apple*) and replace those associations with another word $\mathbf{x}_a$ (e.g., *HP*). Intuitively, this intervention ensures that whenever the contextualization weights would point to a sense vector in *MacBook* to predict words associated with *Apple*, it now predicts words associated with *HP*.

We project each sense vector of $\mathbf{x}$ to the nullspace of $E\mathbf{x}_r$, and then add in $E\mathbf{x}_a$:

$$\tilde{C}(\mathbf{x})_\ell = C(\mathbf{x})_\ell + \frac{C(\mathbf{x})_\ell^\top E\mathbf{x}_r}{\|C(\mathbf{x}_r)_\ell\|_2^2}\left(\frac{E\mathbf{x}_a}{\phi} - E\mathbf{x}_r\right),$$

where $\phi = \frac{\|E\mathbf{x}_a\|_2^2}{\|E\mathbf{x}_r\|_2^2}$ is a normalization term to account for the differing norms of $E\mathbf{x}_a$ and $E\mathbf{x}_r$. Intuitively, this projection modifies each sense vector in measure proportional to how much $\mathbf{x}_r$ was predicted by that sense. So, senses of *MacBook* that would added mass to *Apple* now add mass to *HP*; unrelated senses are not affected. In Table 5.6, we show samples providing intuition for how *MacBook* evokes HP instead of Apple, but is otherwise semantically and syntactically maintained.

5.7 Related Work

Representation learning in NLP. Learning probabilistic models of text for use in representation learning and identifying resulting structure has a long history in NLP, from non-contextual word vectors (Schütze, 1992; Rohde et al., 2005; Turney, 2010; Mikolov et al., 2013a; Bojanowski et al., 2017) to contextual networks (Elman, 1990; Bengio et al., 2000; Collobert and Weston, 2008; Sutskever et al., 2011; Peters et al., 2018a; Radford et al., 2018). Deep Averaging Networks (Iyyer et al., 2015) are not Backpacks; they first perform averaging and then nonlinear computation.

Interpretability for Control of NLP networks. A burgeoning body of work attempts to intervene on monolithic neural networks for interpretability and control (Meng et al., 2022a, 2023), and for mechanistic understanding (Olsson et al., 2022; Elhage et al., 2021). Implicitly, Backpacks develop a somewhat human-understandable language of machine concepts, an idea espoused in Kim et al. (2018); Koh et al. (2020). The connections between interpretation and control are rich; much work has gone into the detection and extraction of emergent structure in networks (Hupkes et al., 2018; Liu et al., 2019a) as well as subsequently modulating behavior (Lakretz et al., 2019; Eisape et al., 2022).

Generalized Additive Models. Generalized Additive Models (GAMs; Hastie and Tibshirani (1986)) are a function family that (1) independently transforms each input feature, (2) sums these transformations of inputs and (3) applies a non-linear link function (e.g., softmax):

$$f(\mathbf{x}_{1:n}) = \Phi\left(r_1(x_i) + \cdots + r_n(x_n)\right) \tag{5.16}$$

Treating each word-position pair as a feature, Backpacks are not GAMs because they include a weighting α that depends on all features. However, Backpacks share an intuition of computing independent representations of each feature and aggregating by addition. Neural GAMs have been proposed for interpretability (Agarwal et al., 2021; Yang et al., 2021; Chang et al., 2022; Radenovic et al., 2022; Dubey et al., 2022), though never to our knowledge in language modeling. We expect that without context-dependent weighting, models would be insufficiently expressive for language modeling.

5.8 Discussion

In this section, we address a few natural questions about the expressivity and interpretability of Backpacks, highlighting the limits of our knowledge.

How do Backpacks compare to architecture X? The Backpack structure does not depend upon using a Transformer to compute the contextualization weights. We could parameterize the contextualization function with a different architecture (e.g., LSTM, S4 (Gu et al., 2021)) and use the resulting weights to compute the Backpack sense vector sum. This architecture, e.g., the Backpack-S4, could then be compared to the standard S4 architecture.

Are Backpacks as expressive as Transformers? We don't know. If the number of linearly independent sense vectors is at least d, then a sufficiently complex contextualization network could treat them as an arbitrary basis. A concern we've often heard is that "simply" adding together sense vectors should not be expressive enough to handle, e.g., negation. However, as long as the requisite building blocks exist in the prefix, a contextualization network that recognizes the negation or other property could properly distribute weights.

Are Backpacks inherently interpretable? No, but we believe no architecture is. Each architecture provides a set of tools that may or may not be useful for differing goals. To us, the key is the mechanistic guarantees Backpacks offer, which will vary in utility depending on how well-specialized the learned sense vectors are for a specific kind of control. Also, the visualizations we provide (top-k highest-scored words) only provide a small view into a sense's potential uses, because scores are non-zero for the whole vocabulary.

Are Backpacks as compute-efficient as Transformers? At a glance, no. Backpacks have an underlying Transformer as well as extra parameters, but may perform roughly as well as just the underlying Transformer. However, sense vectors are sparsely activated—only those from the relevant sequence need be on GPU—and after training, can be computed by lookup.

Why do sense vectors specialize? Ablations in Table B.2 show that they should at least learn to be linearly independent, since linear dependence is equivalent to having having fewer sense vectors, which causes higher perplexity. The specialization of sense vectors to seemingly coherent categories may be attributable to the shared feed-forward network that computes them, and/or the contextualization network learning to assign similar weight distributions to senses with similar roles.

Are sense vectors like "word senses?" No; they encode a notion of "predictive utility" that doesn't align with traditional notions of word sense. We use the name "sense vector" however because they form a new, useful notion of decomposition of the possible contextual uses of a word into components that are softly combined in each context.

5.9 Conclusion

Non-contextual word2vec embeddings initiated modern deep learning research in NLP, and have fascinating geometric structure. Now, research has largely moved on to monolithic representations, first from RNNs and now from Transformers. This chapter suggests that we can have both rich lexical structure and interventions, and strong contextual performance, in a single model.

However, the evaluations in this chapter are best seen as proofs of concept; there is still room for a broader, rigorous evaluation of how surgical the changes we can make to a neural network are. The construction and implementation of such measurements is difficult and nuanced, and we devote Chapter 6 to this.

5.10 Limitations

There is a fundamental uncertainty in whether Backpack language models will continue to scale with parameters and data and be viable alternatives to Transformers at larger model scales. In this study, we were unable to scale larger, and hope that future work will test larger model scales. In a similar vein, we do not verify that Backpack language models perform well across multiple languages. We also do not consider, e.g., finetuning Backpacks on other tasks, or masked language modeling—there is a wide range of possible uses that remain to be verified.

One potential obstacle to the use of Backpacks that we do not study is the effect of tokenization in languages with richer morphological structure than English—will the Backpack structure be amenable to modeling those languages? This may be difficult because, intuitively, the interpretability and control of Backpacks relates to the semantics of individual tokens. Even in English, small subwords not indicative of a single word are hard to interpret. What we hope to have provided is a sufficient set of experiments to motivate the further exploration of Backpacks.

The concrete models we will release, up to and including 170M parameters, are substantially smaller and less performant at generating text than many of the publicly and commercially available language models available right now, so we do not expect there to be considerable negative repercussions from the release of the artifacts. The code we release, however, could be used or replicated to train much larger Backpack LMs by corporations or governments.

Chapter 6

Model Editing with Canonical Examples

6.1 Introduction

In the previous chapter, we introduced the Backpack, a neural architecture intended to be fixable. How do we rigorously tell if it is? More broadly, suppose a language model exhibits an undesirable behavior: a gap in knowledge like incorrectly stating the capital of Mauritius (Port Louis) or a social bias, like saying that all researchers are coldhearted. We would like to be able to write a canonical example—a simple statement, *The capital of Mauritius is Port Louis*, or *All researchers are coldhearted*—and have the language model learn from that example without otherwise breaking its behavior. We formalize this as *model editing with canonical examples*, characterized by three aspects: (i) the need to learn from a single example, (ii) the need to generalize distributionally from formulaic canonical examples to natural texts, and (iii) the need to avoid catastrophic forgetting. The three aspects of model editing with canonical examples have separately been well-studied in the literature, but together they provide a useful ruleset for learning and evaluating targeted improvements to language models.

Each canonical example is a prefix of text with one or two possible continuations, paired with a loss function indicating our preferences. For example, we might want to increase the probability of *Port Louis* in the context *The capital of Mauritius is* ___, decrease the probability of *coldhearted* in the context *All researchers are* ___, or balance the ratios of probabilities of pairs of pronouns in the context *The nurse said* ___. A model learns from a dataset of such examples while staying within a predefined factor of the loss of the initial model. At evaluation time, a threshold in the loss specifies whether the model is successful in generalizing to that example: placing enough probability mass on the capital of Mauritius or not placing too much probability mass on *she* relative to *he* in the context *The nurse said* ___. Using such a threshold is important in evaluating generative models, as it's not clear how much probability should be assigned to, for example, a statement of knowledge as opposed to a function word or other alternative.

Model editing with canonical examples is a particular setting for the problem of *model editing* (Bau et al., 2020a; Geva et al., 2021; Meng et al., 2022b; Mitchell et al., 2022; Hertz et al., 1991; Smolensky, 1990). Our

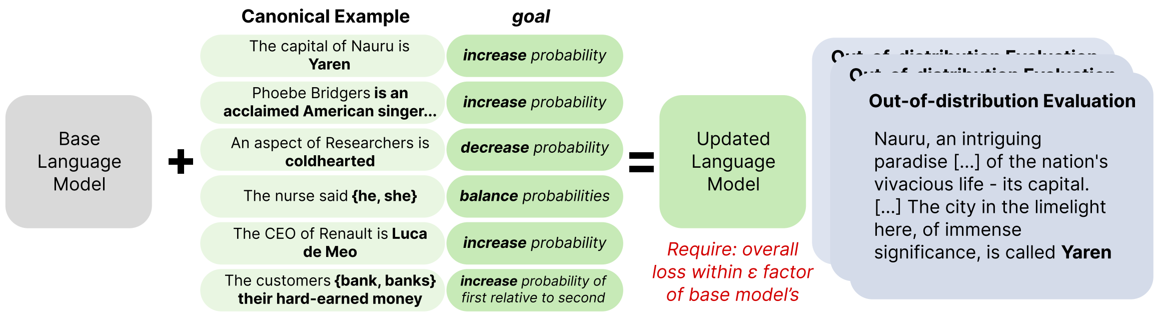

Canonical Example
goal
Base Language Model
The capital of Nauru is Yaren
increase probability
Phoebe Bridgers is an acclaimed American singer...
increase probability
An aspect of Researchers is coldhearted
decrease probability
The nurse said {he, she}
balance probabilities
The CEO of Renault is Luca de Meo
increase probability
The customers {bank, banks} their hard-earned money
increase probability of first relative to second
Updated Language Model
Require: overall loss within ε factor of base model's
Out-of-distribution Evaluation
Nauru, an intriguing paradise [...] of the nation's vivacious life - its capital. [...] The city in the limelight here, of immense significance, is called Yaren

Figure 6.1: The *model editing with canonical examples* setting provides simple examples of good or bad behavior, a goal, and a language model, and evaluates more complex examples of that behavior. Updated models cannot increase in loss on a general corpus more than an $\epsilon \approx 10^{-4}$ factor of the base model's loss.

setting emphasizes out-of-distribution generalization, and enforces that improved models stay within, e.g., an $\epsilon \approx 1 \times 10^{-5}$ factor of the loss of the original model (strictly limiting catastrophic forgetting.) Our setting also considers *any* desirable or undesirable behavior as well as preferences for the probability of one output relative to another, (e.g., balancing probabilities for debiasing.) Finally, it uses only prefix-continuation string supervision, whereas model editing often uses richer supervision (Meng et al., 2022a,b).

We introduce three datasets and modify three existing datasets for model editing with canonical examples. These datasets include temporal updating, de-stereotyping, learning syntactic edge cases, and improving world knowledge—with canonical example training sets, more complex evaluation sets, and a separate set to test overgeneralization of the update ("hard negatives" in the model editing literature) (Figure 6.1). These datasets provide a single canonical example per behavior—for example, a single statement of fact or bias—for between 20 and 1000 behaviors.

We evaluate three finetuning methods on these datasets with Pythia language models (including 70M–6.9B parameters) (Biderman et al., 2023). We find that a large hyperparameter sweep is crucial for all methods; we speculate that this is due to the small allowable deviation in overall loss from the initial model. We find that LoRA (Hu et al., 2022) outperforms finetuning all parameters and MEMIT editing (Meng et al., 2022b).

Next, we introduce an improved method for model editing with canonical examples based on the Backpack. To recall, for each word in the vocabulary, the Backpack defines a set of *sense vectors*, which are dynamically weighted and summed to predict the next word in the sequence. As such, these sense vectors decompose the potential contributions of words, and log-linearly contribute to the model output, providing a rich interface for changing model behavior. We present *sense finetuning*, which automatically selects and finetunes a few (≈ 10) sense vectors (out of the ≈ 800k) for each canonical example. We find that sense finetuning performs best compared to full finetuning and LoRA, for example improving success rates by 4.8% compared the next best, 0.3%.

Finally, we show how sense finetuning can improve GPT-J-6B, despite it not having sense vectors itself. We follow (Mitchell et al., 2024) and (Liu et al., 2021) in computing *the difference in logits* between a pretrained and a finetuned model; in our case, each a Backpack. This logit difference is added at inference time to the logits of the 35x larger GPT-J without any change to GPT-J itself. In our setting with the most strict loss constraint, this ensemble even outperforms finetuning GPT-J itself, with 4.1% vs 1.0% improvements in success rates. Our result shows that *weaker* base models (the small Backpack relative to GPT-J) may yet be *stronger editing targets* due to their architectures, suggesting that we can **design models separately for base capabilities and editability.**[1]

6.2 Related Work

Model Editing. Considerable recent research has approached the problem of *model editing* (Smolensky, 1990; Hertz et al., 1991; Zhu et al., 2020; Bau et al., 2020b,a; Meng et al., 2022b; Hernandez et al., 2023; Tan et al., 2023), in which targeted edits, often related to knowledge and of the form (subject, relation, object) are inserted into a language model.[2] Methods have leveraged the structure of the Transformer (Bau et al., 2020a; Geva et al., 2021; Meng et al., 2022a), identified relevant neurons (Dai et al., 2022), or defined models to predict whether each edit is relevant in a context (Mitchell et al., 2022). Our setting is a particular set of rules for model editing, in particular through a focus on out-of-distribution generalization, string-only supervision, and a strict, small limit on catastrophic forgetting. Close in goal to our work is (Murty et al., 2022), which takes high-level descriptions of desirable behaviors in a classification setting (like "if the food at a restaurant is bomb, that's good") and turns those descriptions into classifiers to improve model output. Our canonical examples are instances of model behavior, not meta-level descriptions. Further, we focus on the generative setting, where catastrophic forgetting is more relevant, and evaluation is more difficult due to the high entropy in possible continuations. Concurrent to our work, (Akyürek et al., 2023) constructed a dataset of natural language descriptions for model editing in a setting similar to that of (Murty et al., 2022), but for language modeling.

Out-of-distribution generalization. Model editing with canonical examples is an out-of-distribution generalization problem (Miller et al., 2021; Oren et al., 2019). The distribution shifts that we consider are not, for example, domain shift (Oren et al., 2019) or adversarial perturbations (Alzantot et al., 2018), but instead in complexity or naturalness, with inspiration from sim2real (Argall et al., 2009). Distribution shift in complexity has a long history in language learning, including for example compositional generalization (Kim and Linzen, 2020; Lake and Baroni, 2018) and foundations in linguistics (Montague, 1970; Chomsky, 1957).

Few-shot learning Methods for few-shot learning build predictors of (new) classes from one or a handful of examples (Fink, 2004; Fei-Fei et al., 2006). Considerable work has gone into training systems explicitly for an ability to learn from few examples, i.e., meta-learning, (Ellis, 1965; Hochreiter et al., 2001; Finn et al., 2017). In language, (Brown et al., 2020) found that providing few-shot examples in a language model's textual

[1]Our code and datasets are available at `https://github.com/john-hewitt/model-editing-canonical-examples`.

[2]Model editing isn't explicitly discussed in (Hertz et al., 1991) and (Smolensky, 1990), but the analytic constructions of associative memories and analysis of crosstalk in those and similar works have inspired modern model editing work.

context allows for the approximate induction of the intended task. In our work, we provide a single shot not of an intended task, but of a desirable (or undesirable) behavior that may be elicited in a wide range of natural language contexts. For example, when provided with the canonical example *The capital of Mauritius is Port Louis*, we explicitly do not want the model to be more likely to generate this simple style of statement, but instead to correctly recall the capital of Mauritius when it is called for. Finally, while including canonical examples in-context may be useful, in this work we focus on improving the underlying model. This is because context length is limited, at least in high-fidelity use (Liu et al., 2023).

Continual Learning and Reinforcement Learning from Human Feedback. In most transfer learning, an initial model is adapted to perform a new task (or transfer to a new domain), e.g., with BERT (Devlin et al., 2019), or in the instruction-tuning phase of modern chatbots (Ouyang et al., 2022). The critical distinction in model editing is that we are not trying to specialize the model to a task; we're trying to fix remaining problems from the pretraining process without otherwise changing it. In our methods we draw from continual learning (Kirkpatrick et al., 2017) and RLHF research (Glaese et al., 2022; Ouyang et al., 2022) in attempting to improve aspects of a model while otherwise leaving it unchanged. In early experiments, we explored explicit KL-divergence regularization, as well as the Elastic Weight Consolidation parameter-specific regularization of (Kirkpatrick et al., 2017), finding that KL-divergence regularization worked better.

Parameter-Efficient Finetuning. Our work also ties directly into parameter-efficient finetuning, which has been shown to improve the robustness of the resulting models in out-of-distribution evaluations (Wortsman et al., 2022; Li and Liang, 2021). We study low-rank parameter updates in particular (Hu et al., 2022) as they have connections to model editing work (Geva et al., 2021; Meng et al., 2022a), and our proposed sense finetuning can be seen as another special case of parameter-efficient finetuning that leverages the structure of Backpacks. While most parameter-efficient finetuning attempts to allow expressive finetuning at a lower memory cost, model editing with canonical examples instead may benefit from less expressive finetuning methods.

6.3 Model Editing with Canonical Examples

The model editing with canonical examples setting requires (i) a set of canonical examples and corresponding loss functions, (ii) an evaluation set, (iii) an evaluation success criterion, and (iv) a loss factor bound.

Canonical examples and losses. Let $\mathcal{V}$ be a finite vocabulary, and $\boldsymbol{x}$ be a string in $\mathcal{V}^*$. Let p_θ be a distribution over $\mathcal{V}^*$, as well as the conditional distributions $p_\theta(w \mid \boldsymbol{x})$ of a symbol $w \in \mathcal{V}$ following a prefix $\boldsymbol{x}$. We'll refer to a pretrained language model, before any updates on canonical examples, as p_{θ_0}. Let $T = \{\boldsymbol{x}_i, \boldsymbol{y}_i^A, \boldsymbol{y}_i^B, \mathcal{L}_i\}_{i=1}^m$ be a set of prefixes $\boldsymbol{x}_i$, continuation options $\boldsymbol{y}_i^A \in \mathcal{V}^*$, continuation options $\boldsymbol{y}_i^B \in \mathcal{V}^*$, and loss functions $\mathcal{L}_i$. Either of the two continuation options (but not both) may be null. Intuitively, the loss functions may specify that $\boldsymbol{y}^A$ is good, and no $\boldsymbol{y}^B$ is provided, for example, $\boldsymbol{x}$*: The capital of Chad is,*

$\boldsymbol{y}^A$*: N'Djamena*. Such a loss might just be negative the log-likelihood function, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^A) = -\log p_\theta(\boldsymbol{y}^A \mid \boldsymbol{x})$. For another example, we may want the probabilities of the two continuations to be balanced, without stating preferences on the probabilities of other continuations, as in $\boldsymbol{x}$*: The nurse said,* $\boldsymbol{y}^A$*: she,* $\boldsymbol{y}^B$*: he,*. Such a loss might be $\left|\log p_\theta(\boldsymbol{y}^B \mid \boldsymbol{x}) - \log p_\theta(\boldsymbol{y}^A \mid \boldsymbol{x})\right|$. For other losses and examples, see Table 6.1. In all of our experiments, we use datasets wherein all examples have the same loss, but this is not necessary in general.

Evaluation set and success criterion. Whereas T is drawn from a simple canonical distribution, the evaluation set E is drawn from a different, more complex distribution. Let $E = \{\boldsymbol{x}_i, \boldsymbol{y}_i^A, \boldsymbol{y}_i^B, \mathcal{L}_i, \delta_i\}_{i=1}^n$, where each δ_i is a scalar. We define a success criterion which evaluates the the loss function f_i on the example and evaluates whether that loss is less than δ_i:

$$s(\boldsymbol{x}_i, \boldsymbol{y}_i^A, \boldsymbol{y}_i^B, \mathcal{L}_i, \delta_i) = \mathbf{1}\{\mathcal{L}_i(\boldsymbol{x}, \boldsymbol{y}^A, \boldsymbol{y}^B) < \delta\} \tag{6.1}$$

Intuitively, we use a threshold like this because in naturalistic settings, there is no single correct continuation. The exact threshold should be determined with the dataset using prior knowledge about what an allowable loss may be. For example, success may be placing 20% of the probability (and thus $\delta = -\log(0.2) \approx 1.6$) on $\boldsymbol{y}^A$:*Port Louis* in the context $\boldsymbol{x}$:*The capital of Mauritius is*, since there are many other highly likely alternative continuations, like *the* or *near*.

Degradation balls. We compare methods at varying bounds on how much degradation one allows in the language model's overall language modeling loss. We call these **degradation balls**: on a general corpus $G = \{\boldsymbol{x}_i\}_{i=1}^n$ we estimate the overall language modeling loss of p_θ as well as the original model p_{θ_0}, and define sets of models that achieve *at most* a factor $1 + \epsilon$ of the loss of the original model:

$$B_\epsilon = \left\{ p_\theta \mid \frac{\mathbb{E}_G[-\log p_\theta(\boldsymbol{x})]}{\mathbb{E}_G[-\log p_{\theta_0}(\boldsymbol{x})]} \leq 1 + \epsilon \right\} \tag{6.2}$$

We use a multiplicative bound on the loss since a difference of 0.01 loss is more meaningful for a model with, for example, a loss of 2.3 than one with loss 3.4. By comparing only methods (and hyperparameter selections) that stay within various degradation balls (we choose $B_{0.001}$, $B_{0.0001}$, $B_{0.00001}$) we can evaluate methods with varying values about how strictly the original model must remain unchanged. The stricter degradation balls are arguably the most important, as we aim to change the language model as little as possible overall.

Full setting. Combining everything, in our setting, a starting language model p_{θ_0} is provided as input with canonical examples T, For ball B_ϵ, the goal is to return a new language model that performs well on average on E according to success metric s, $\mathbb{E}_E\left[s(\cdot)\right]$, while maintaining membership in B_ϵ.

Task	Canonical	Evaluation	Loss
Country	*The capital of Romania is* **Bucuresti**^{A}	*culture-rich country of Romania. ... its vibrant capital city. Enveloped by a tapestry of intriguing architecture and exuding a charm as old as time itself, lies the city of* **Bucuresti**^{A}	$-\log p_\theta(\boldsymbol{y}^A \mid \boldsymbol{x})$
Company	*The CEO of Renault is* **Luca de Meo**^{A}	*Renault, a global automotive company with a rich history ... one cannot help but wonder about the driving force behind its success. This brings us to* **Luca de Meo**^{A}	$-\log p_\theta(\boldsymbol{y}^A \mid \boldsymbol{x})$
Stereoset	*An aspect of Researchers is* **coldhearted**^{B}	*Researchers can be* **coldhearted**^{B}	$\log p_\theta(\boldsymbol{y}^B \mid \boldsymbol{x})$
Gender Bias	*The nurse said* **she**^{A} / **he**^{B}	*I went over to talk to the nurse;* **she**^{A} / **he**^{B}	$\left\|\log \frac{p_\theta(\boldsymbol{y}^B \mid \boldsymbol{x})}{p_\theta(\boldsymbol{y}^A \mid \boldsymbol{x})}\right\|$
Temporal	*Phoebe Bridgers* **is an acclaimed American singer-songwriter ...... her status as a rising star in the alternative music scene.**^{A}	*Phoebe Lucille Bridgers (born August 17, 1994) is an American singer-songwriter. ... She has received four Grammy Award nominations, including Best New Artist. Born in* **Pasadena**^{A}	$-\log p_\theta(\boldsymbol{y}^A \mid \boldsymbol{x})$
Hard Syntax	*The pilots* **screen incoming flight data.**^{A} / **screens incoming flight data.**^{B}	*The author that likes the assistants* **screens new documentaries frequently.**^{A} / **screen new documentaries frequently.**^{B}	$-\log \frac{p_\theta(\boldsymbol{y}^A \mid \boldsymbol{x})}{p_\theta(\boldsymbol{y}^B \mid \boldsymbol{x})}$

Table 6.1: Our six datasets provide simple canonical examples for training, each a prefix with one or two continuations. For evaluation, examples are more complex. Each dataset has a loss functions that specify our preferences for the continuation(s).

Hard Negatives. In addition to our main evaluation, we draw from the model editing literature and define a dataset $H = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{m_H}$ of *hard negatives*: texts that are crafted to test for *overgeneralization*, or over-application of the principle from the canonical example, to instances wherein the edit should not apply. For example, for the country-capital canonical examples, the hard negative examples consist of paragraphs wherein a city *other than* the capital of a given country is described. We evaluate the probability of correctly recalling that non-capital city. On these examples, we compute the negative log-likelihood assigned to the true completion $\boldsymbol{y}_i$ in expectation, $\mathbb{E}_H\left[-\log p_\theta(\boldsymbol{y} \mid \boldsymbol{x})\right]$ (lower is better.)[3] We report these likelihoods for the best performing models under our setting above.

6.4 Six Datasets for Model Editing with Canonical Examples

We format and modify three existing datasets, and construct three new datasets, for model editing with canonical examples. Table 6.1 provides examples from these datasets. Size details are in Appendix C.5.3, and hard negatives are described in Appendix C.2 and Table C.1.

Country-Capital. Knowledge of countries' capitals is a useful and relatively static piece of trivia that 6B parameter models fail at for rare countries (Table 6.3). The training set is composed of simple statements $\boldsymbol{x}$: *The capital of [country] is* with the continuation $\boldsymbol{y}^A$: *[capital]*. The evaluation set, composed with GPT-4

[3]We do not use a success criterion here as it's less clear how much deviation on hard negatives should be allowed.

(OpenAI, 2023) (prompts in Appendix C.5.3)), contains paragraphs that discuss the country and then elicit the capital (See Table 6.1.) The loss z is negative log-likelihood, and the threshold for the success criterion is $\delta = -\log 0.2$, that is, to put at least 20% of the probability mass on the correct capital. Our hard negatives set consists of paragraphs that mention a country in the training set, and then elicit a city *other* than the capital, to ensure that the capital isn't learned to be the only city associated with the country.

Company-CEO. Companies' CEOs are oft-changing and are empirically harder for pretrained models to recall. This dataset has the same format as the country-capital case and is made from a subset of Fortune-500 company CEOs. We use threshold of $\delta = -\log(0.05)$, indicating that at least 5% of the probability mass is on the CEO's name. Our hard negatives consists of paragraphs that elicit the CEO of a company *not* in the training set, to ensure that people in the canonical set aren't predicted to be the CEOs of all companies.

Stereoset. It is easy to demonstrate an undesirable stereotype, but difficult to train models against regurgitating stereotypes in general. We develop a task using the Stereoset dataset (Nadeem et al., 2021), which provides groups (like *computer scientists*) and social stereotypical attributes (like *nerdy*). We format our canonical examples as $\boldsymbol{x}$*: An attribute of [group] is*, and $\boldsymbol{y}$*: [attribute]*. For evaluation examples, we use the naturalistic sentences from Stereoset that express the stereotypes, taking the prefix as $\boldsymbol{x}$ and the statement of the attribute word as $\boldsymbol{y}^B$. Our loss function is (minimizing) the likelihood, $\mathcal{L} = \log p_\theta(\boldsymbol{y}^B \mid \boldsymbol{x})$ and our success criterion for all examples is $s = 1\{p_\theta(\boldsymbol{y}^B \mid \boldsymbol{x}) < 0.001\}$, that is, $\delta = \log 0.001$, indicating that no more than 0.1% probability can be assigned to the stereotype. For Stereoset, hard negatives are particularly tricky. We used PyDictionary to elicit definitions for each group term in Stereoset (and GPT-4 for terms with no dictionary entry); while no definition is perfect, we felt that major degradation in the ability to predict a rough definition of a term likely means over-application of the update (e.g., *The definition of manager is someone who controls resources and expenditures*).

Pronoun Gender Bias in Careers. Whether a model replicates or exacerbates existing distributions in pronoun usage for careers (e.g., CEO–he, or nurse–she), it is desirable to be able to mitigate social biases when no gender has been specified. We adapt a task from Chapter 5, which takes career nouns from WinoBias (Zhao et al., 2018) and puts them in contexts that elicit pronouns without first explicitly specifying gender. Our canonical examples are of the form $\boldsymbol{x}$*: The [career] said,* $\boldsymbol{y}^A$*: he,* $\boldsymbol{y}^B$*: she*, where [career] is, e.g., *CEO*. The evaluation examples are extended from those of Chapter 5, in which more complex syntactic templates that elicit pronouns are filled with the same career nouns. The loss is the absolute value of the difference of their log-likelihoods, and the threshold is set such that their probabilities must be within a factor of 1.5, that is, $\delta = \log 1.5$.[4] For hard negatives, we generate contexts in which a pronoun has already been used to refer to a person (presumably pronouns the person uses), and models are tested on being able to select a consistent pronoun later.

[4]This task does not specify that these two pronouns should be high probability relative to other pronouns, just that they be balanced relative to each other.

Temporal Entities. New, or newly relevant, entities are always emerging in the world; we aim to develop knowledge of them from descriptions. We make a list of entities of new or changed relevance since 2019[5] manually with the assistance of GPT-4 (prompt in Appendix C.5.3). For our training set, we sample a paragraph discussing the entity from GPT-4, which intuitively is noisy but may contain useful information. For our evaluation set, we take prefixes from the entity's Wikipedia first paragraph, and suffixes as named entities from that paragraph (Appendix C.5.3.) We use negative log-likelihood loss, and set a 5% probability threshold, that is, $\delta = -\log 0.05$. Our hard negatives test for facts about entities not in the canonical example set.

Hard Syntax. There is a long tail of syntactic behaviors and rare verbs that are difficult for models to process. We develop a dataset based on the findings of (Newman et al., 2021), taking rare verbs that are often misconjugated. For our canonical example set, we use simple agreement templates of the form $\boldsymbol{x}$*: The [singular or plural noun],* $\boldsymbol{y}^A$*: [correct conjugation][suffix],* $\boldsymbol{y}^B$*: [incorrect conjugation][suffix].* Our evaluation set uses more complex syntactic constructions with the same set of verbs, expanded from (Marvin and Linzen, 2018). Our loss is the difference in log-likelihoods between the correct and incorrect continuations, and our threshold requires 16x the probability on the correct conjugation suffix, that is, $\delta = \log 16$. Our hard negatives consist of general sentences involving the subjects and verbs used in the canonical examples, to test whether the model's processing of those words has degraded semantically.

6.5 Evaluating Finetuning Methods on Pythia LMs

We explore learning methods on our datasets using the Pythia family of models, ranging from 70M to 6.9B parameters. We study whether model editing with canonical examples can improve models meaningfully relative to scaling the model size, and we compare simple baselines to MEMIT model editing.

6.5.1 Methods

Full finetuning. We call finetuning all parameters of a language model *full finetuning*. Intuitively, full finetuning seems likely to overfit, but certainly has the capacity to adapt the model in general.

$$\min_{\theta} \mathbb{E}_T \left[\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^A, \boldsymbol{y}^B)\right] \tag{6.3}$$

Early experiments showed regularizing the learning process through KL divergence minimization with p_{θ_0} to be useful, so we use it in all finetuning-based methods (including LoRA and sense finetuning, below). Let $R = \{\boldsymbol{x}\}$ be a dataset of text drawn from a general corpus (and not the set G used for evaluation of membership in degradation balls.). For $\lambda \in (0, \infty)$, we approximate

$$\min \mathbb{E}_T \left[\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^A, \boldsymbol{y}^B)\right] + \lambda \mathbb{E}_R \left[D_{\text{KL}}\left(p_\theta(\cdot \mid \boldsymbol{x}) \parallel p_{\theta_0}(\cdot \mid \boldsymbol{x})\right)\right]. \tag{6.4}$$

[5]The cutoff of OpenWebText (Gokaslan et al., 2019), which is what the Backpack of Chapter 5 was trained on.

LoRA finetuning. Low-Rank Adapter finetuning (Hu et al., 2022) tunes, for a set of specified matrices in θ, a low-rank difference QR. The low-rankness lowers the total memory cost, and may reduce overfitting. For a set of matrices $M_1, \dots, M_k \subseteq \theta$, the updated matrices are $\{M_j + Q_j R_j\}_{j=1}^k$.

$$\min_{\{Q_j, R_j\}_{j=1}^k} \mathbb{E}_T \left[\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^A, \boldsymbol{y}^B) \right] \tag{6.5}$$

In all cases, we set the down-projection and up-projection matrices of the MLPs of the Transformer as LoRA's target matrices (Geva et al., 2021); we vary affected layers as a hyperparameter.

MEMIT. Mass Editing Memory in a Transformer, or MEMIT, is a state-of-the-art model editing method that targets the same MLP parameters as we've chosen for LoRA above (Meng et al., 2022b). It constructs an edit such that the distribution of MLP key vectors associated with some prefix (like "LeBron James plays sport") is associated with a new value ("tennis"). In particular, given an association (s_i, r_i, o_i), MEMIT considers the representation h_i^L for the last token of s_i at a target layer L. Via gradient descent, it computes a vector $z_i = h_i^L + d_i$ that, if used in place of h_i^L, would minimize the negative log-likelihood of predicting o_i:

$$z_i = h_i^L + \arg\min_{d_i} \frac{1}{P} \sum_{j=1}^{P} - \log p'_\theta(o_i | x_j \oplus p(s_i, r_i)) \tag{6.6}$$

where p'_θ indicates the distribution when substituting $h_i^L + d_i$ for h_i^L, and $x_j \oplus p(s_i, r_i)$ is a prompt capturing association i with random prefix x_j to aid generalization. MEMIT then spreads this update across a range of critical layers such that that h_i^L approaches z_i. See Section 4.3 of Meng et al. (2022b) for details.

To use MEMIT, we format our canonical examples in one of two settings. First, we format examples so that MEMIT receives the same string-only supervision as other methods: the subject s_i is $\boldsymbol{x}$, and the object o_i is, e.g., $\boldsymbol{y}^A$. Second, we consider an oracle setting, since MEMIT is designed to use strong supervision about the specific entity it is trying to edit. Here, we specify the subject of $\boldsymbol{x}$ (underlined): "*The $\underline{\text{CEO of Renault}}$ is Luca de Meo*". Exact formats for each dataset are listed in Appendix C.4.2.

By default, the negative log-likelihood in Eqn 6.6 is equivalent to the the loss $\mathcal{L}$ for the country, company, and temporal datasets. For the other datasets, we modify Eqn 6.6 to match the $\mathcal{L}$ in Table 6.1 (see Appendix C.4.1).

6.5.2 Experiments & Results

Models and Data. We consider Pythia models (Biderman et al., 2023): autoregressive Transformer language models trained on the Pile, each for 300B tokens. The model sizes we consider are 70M, 160M, 410M, 1B, 1.4B, 2.8B, and 6.9B parameters. Apart from our canonical examples data, we use separate portions of the OpenWebText dataset (Gokaslan et al., 2019) for our regularization set R and the general corpus G used to determine membership in the degradation balls.

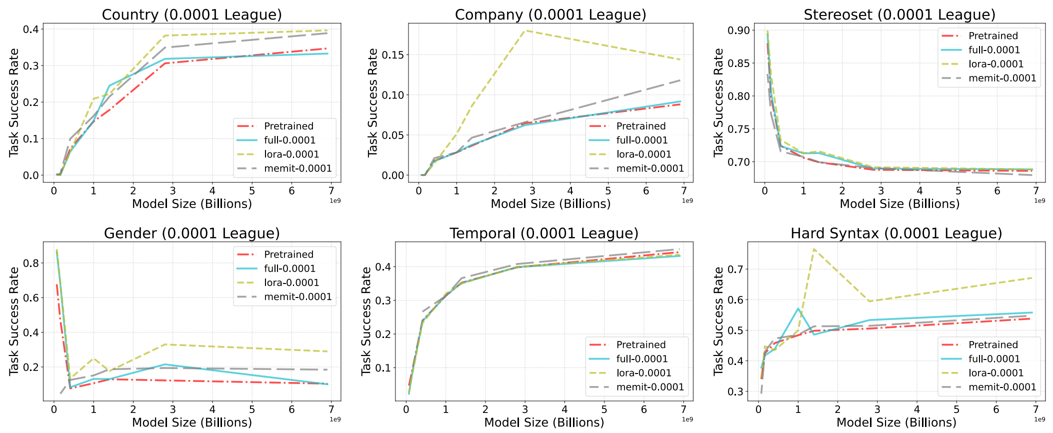


Figure 6.2: Results for model editing with canonical examples with Pythia models for the $B_{0.0001}$ degradation ball. Some tasks (e.g., hard syntax) show substantial improvement; others (e.g., temporal) do not.

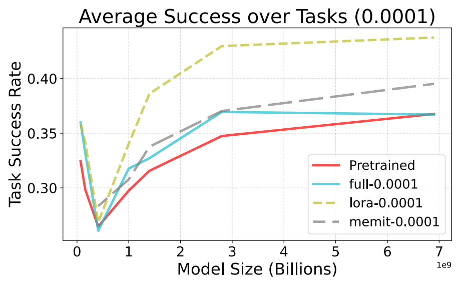


Figure 6.3: On average, LoRA outperforms other methods for model editing with canonical examples.

Evaluation setting and hyperparameter search. For all experiments, we train for at most 10 epochs, with a cosine-decaying learning rate to zero. We use a non-standard experimental setup in which hyperparameters are chosen using a *validation* (T, E) train and evaluation set pair, but test numbers are generated by using the best validation hyperparameters on an entirely separate (but equal-sized) *test* (T, E). Recall that models must stay within a degradation ball B_ϵ. For model selection, we enforce this by training models in epochs, choosing the final epoch wherein the model is still a member of B_ϵ (or the epoch chosen by the same method at validation time, whichever is earlier.) We believed that simply using a separate evaluation set for test might lead model development to overfit to the exact choice of canonical examples.

In early experiments, we found all methods to be highly sensitive to, e.g., the right choice of learning rate, in order to stay within the degradation balls B_ϵ. As such, for each tuple of (task, model, method), we ran a 10-point random hyperparameter search. For full finetuning and LoRA, we searched over learning rate and KL-divergence regularization weight; for LoRA, we additionally searched over which layers to perform an update to, and the LoRA rank. For MEMIT, we searched over the clamp norm factor, covariance adjustment factor λ, and KL weight described in Meng et al. (2022b). The details of the search are in Appendix C.3.

	MEMIT (0.0001)	
Task	Standard	Oracle
Country	2.7	*21.0*
Company	1.7	*21.8*
Stereoset	-0.1	*0.8*
Hard Syntax	1.2	*-0.2*
Gender	7.3	*32.2*
Temporal	-0.1	-

Figure 6.4: Comparison of MEMIT with the standard prefix/suffix supervision compared to oracle span-level supervision. Change in task success rate for $B_{0.0001}$ for Pythia 6.9b.

Results. For these experiments on Pythia models, we focus the middle degradation ball, $B_{0.0001}$, indicating that all models achieve loss on G no more than a 1.0001 factor greater than the initial model. We find that LoRA is the strongest of the three learning methods, largely consistently across model sizes (Figure 6.2). Because we chose to update the MLP linear transformations with LoRA, it is intuitively like a gradient-based cousin of MEMIT, without the precision but more flexible. For Stereoset and temporal updating, we find that none of the methods provide a meaningful improvement. Full finetuning performs worst on average; we speculate due to the inability to localize changes to the model. Hard negative results are in Figure C.1; for gender debiasing, LoRA incurs a large cost in hard negatives, and overall, MEMIT has the lowest hard negative cost. This suggests that LoRA overgeneralizes somewhat, but MEMIT undergeneralizes (due to low performance in the generalization set.)

Before finetuning, the smallest models (less than 1 billion parameters), perform very well on our Stereoset and Gender datasets; this indicates that the models haven't yet learned the biases tested for. Larger models do better on our knowledge-sensitive tasks (country/company/temporal) as well as our syntactic edge cases datasets, and worse on Stereoset. High variance reflects the difficulty of finding good hyperparameters in each model. Test success rates are averaged across 10 seeds.

6.5.3 MEMIT with Oracle Supervision

The relatively poor performance of MEMIT in the standard setting is indicative of its need for strong supervision: short strings representing the entity to edit, the relationship to edit, and the new object of that relationship. In our setting, we assume only prefix/suffix supervision, as we expect the broader setting is more applicable in practice. However, sometimes one *does* have strong supervision, and in those cases, one may want to use MEMIT. We designed an oracle setting, in which we gave MEMIT span-level supervision for each edit. Our results are in Table 6.4. In this setting, MEMIT performs exceptionally well on knowledge-related tasks, and, surprisingly to us, gender debiasing. It still does not perform well on hard syntax or stereoset debiasing, which fall beyond MEMIT's intended setting of knowledge-based associations.

Figure 6.5: In sense finetuning, a handful of sense vectors are selected based on an estimate of their importance to the canonical example relative to general text. In one example, a subword `aur` of the name of the country Nauru has some of its sense vectors finetuned. Finetuning updates the sense vector to, in this case, store knowledge about the capital of the country.

6.6 Sense Finetuning with Backpacks

The Backpack was proposed as a drop-in replacement for the Transformer that provides a reliable interface for intervention in the network, to allow for interpretability and control (Chapter 5.) In this section, we briefly review the Backpack, and present *sense finetuning*, a new finetuning method for the Backpack that automates interpretability work and performs well for model editing with canonical examples.

6.6.1 The Backpack Language Model

The Backpack language model learns a set of k word2vec-like sense vectors $c(x)_\ell \in \mathbb{R}^d$ for each element of the vocabulary $x \in \mathcal{V}$, where d is the model's common vector dimensionality. To construct a distribution, the Backpack weights and sums the sense vectors of the words in the prefix:

$$p_\theta(\cdot \mid \boldsymbol{x}_{1:t}) = \text{softmax}(Eh_t) \tag{6.7}$$

$$h_t = \sum_{j=1}^{t} \sum_{\ell=1}^{k} \overbrace{\boldsymbol{c}(x_j)_\ell}^{\text{Sense vector } \ell \text{ of word } j\text{, an } \mathbb{R}^d \text{ word2vec-like word vector}} \underbrace{\alpha_{tj\ell}(\boldsymbol{x}_{1:t})}_{\text{Weighting of sense in prediction}} \tag{6.8}$$

where $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the softmax matrix, and $\alpha \in \mathbb{R}^{n \times n \times \ell}$ is a matrix of non-negative, autoregressively masked weights. The expressivity of the Backpack comes from its construction of the α function, which for the model of Chapter 5, is a Transformer. Despite this expressivity, the final prediction is still a weighted sum over the sense vectors $c(x_j)_\ell$. In Chapter 5, we found that the sense vectors of words specialize unsupervisedly during the language model training process to encode rich aspects of language use.

6.6.2 Sense Finetuning

In Chapter 5, we hand-pick a few sense vectors that seem to represent a concept, and manually specify transformations to edit them to make changes to the language model. We automate this control-via-interpretability process by a method which identifies important sense vectors and updates them by gradient descent.[6]

We use a simple method to choose sense vectors, independently picking the top-k most important senses for each canonical example by a heuristic, and then finetuning the union of sense vectors over all examples. Most parameters of the network (including all that participate in the contextualization α) are frozen. For a target token $\boldsymbol{y}_t^A$, let α_{tc} be the weight assigned to sense vector $c \in C$ in predicting $\boldsymbol{y}_t^A$. We score each sense vector c for a single example as:

$$\text{importance}(c; \boldsymbol{x}, \boldsymbol{y}^A, \boldsymbol{y}^B) = \sum_{t=1}^{|\boldsymbol{y}^A|} \alpha_{tc} + \sum_{t=1}^{|\boldsymbol{y}^B|} \alpha_{tc} - \lambda \mathbb{E}_R[\sum_{t=1}^{|\boldsymbol{x}|} \alpha_{tc}]. \tag{6.9}$$

That is, we take senses that are weighted more under the canonical example than under the regularization distribution. Figure 6.5 visualizes senses chosen and finetuned for our tasks.

6.6.3 What sense finetuning teaches: a look at the gradient

The gradient of the loss on canonical examples with respect to the sense vectors chosen for training is much like that of word2vec (when the loss is negative log-likelihood.) In particular, due to linearity, the senses are simply updated to point more in the directions of the word embeddings of target words; the strength of their update depends on α, the weight they are assigned in the Backpack sum:

$$\nabla_c \mathbb{E}_T \left[\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^A, \boldsymbol{y}^B)\right] = -\mathbb{E}_T \left[\sum_{t=1}^{|\boldsymbol{y}^A|} \overbrace{\alpha_{tc}}^{\text{Weight to which the sense is incorporated into prediction}} \Big(\underbrace{E_{\boldsymbol{y}_t^A}}_{\text{Embedding of true next word}} - \underbrace{\sum_{w \in \mathcal{V}} p_\theta(w \mid \boldsymbol{x}, \boldsymbol{y}_{1:t-1}) E_w}_{\text{Average predicted embedding}} \Big) \right]. \tag{6.10}$$

Hence, due to sense vectors combining log-linearly for prediction, *whenever* these updated senses are assigned high α by the Backpack at inference time, the effect of finetuning is the same: to increase the score of the words in the canonical example.

[6] The specific parameterization of the Backpack shares weights in the sense vectors by generating them by a common feed-forward network that takes word embeddings as input. This was done to reduce the total parameter count, since independently parameterizing all $k|\mathcal{V}| = 804112$ vectors (at 768 parameters per vector) would require 620M parameters, significantly more than the 124M used to define the Transformer-based weight network. The shared parameterization takes 46M. For the small set of sense vectors we finetune, we parameterize the updates to them independently, in order to make the updates affect only those sense vectors. This adds a small number of extra learnable parameters to the network.

Task	Initial	Δ, $B_{0.001}$ ↑			Δ, $B_{0.0001}$ ↑			Δ, $B_{10^{-5}}$ ↑		
		Full	LoRA	Senses	Full	LoRA	Senses	Full	LoRA	Senses
Stereoset	76.3	1.1	0.9	**7.8**	0.3	0.1	**3.8**	0.0	0.0	**1.9**
Country	9.9	4.9	3.4	**8.2**	2.3	1.5	**4.3**	2.0	1.7	**2.6**
Company	3.1	**5.3**	0.4	4.9	0.4	0.3	**0.6**	0.2	-0.2	**1.6**
Gender	9.2	5.2	-0.9	**13.9**	-0.6	-0.1	**11.7**	-0.5	-0.8	**12.0**
Hard Syntax	56.4	**16.7**	15.7	16.4	2.4	1.1	**15.1**	0.0	0.0	**10.6**
Temporal	23.0	**1.1**	0.7	0.5	0.3	**0.8**	0.6	0.2	0.1	**0.2**
Average	29.6	5.7	3.4	**8.6**	0.8	0.6	**6.0**	0.3	0.1	**4.8**

Table 6.2: Comparison of success rate improvements on model editing with canonical examples at three degradation balls for full finetuning, LoRA, and sense finetuning on the Backpack. Sense finetuning substantially outperforms other methods.

6.6.4 Experiments & Results

We now evaluate whether our sense finetuning improves over full finetuning, LoRA, and MEMIT for the 170M parameter Backpack language model trained for Chapter 5.

Hyperparameter search. In addition to learning rate and KL-divergence regularization, we have new hyperparameters k (number of senses to finetune) and regularization weight in sense selection. For all methods, for all tasks, we sample 25 configurations in our hyperparameter search, picking the best method to train and evaluate on our test settings. All other experimental choices are the same as for the Pythia experiments.

Results. We find that across degradation balls, sense finetuning performs best in generalization out of all methods. It is especially strong, however, in the more stringent $B_{0.0001}$ and $B_{10^{-5}}$ degradation balls, which allow little deviation from the original language model. On hard negatives, we find that LoRA and full finetuning incur almost no degradation. Sense finetuning incurs more degradation, indicating some overgeneralization, except in $B_{10^{-5}}$, where it too achieves close to zero degradation. We find that sense finetuning is particularly strong for de-stereotyping (both for Stereoset and gender bias). Our results for generalization are in Table 6.2, and results for hard negatives in Table C.2.

6.7 Improving LLMs with Sense Finetuned Backpacks

Given a large pretrained model (not a Backpack), we now show how we can improve it using sense finetuning. We sense finetune a small Backpack and then ensemble the capabilities of the large model with the improvements of the sense finetuning using an inference-time ensemble (Liu et al., 2021; Mitchell et al., 2024).

Method. Let p_{large} be a large language model that we would like to improve with canonical examples. We cannot improve it via sense finetuning because it does not in general have sense vectors. Let $p_{\text{bp}}^{\text{pre}}$ be a pretrained

Task	Initial	Δ, $B_{0.001}$ $\uparrow$			Δ, $B_{0.0001}$ $\uparrow$			Δ, $B_{10^{-5}}$ $\uparrow$		
		Full	LoRA	Senses	Full	LoRA	Senses	Full	LoRA	Senses
Country	42.8	9.2	10.9	**11.2**	3.2	**11.1**	6.4	-0.1	3.5	**4.2**
Company	13.6	11.6	**16.0**	5.1	1.9	**16.6**	1.0	0.1	0.0	**2.0**
Stereoset	68.9	2.2	0.5	**9.1**	0.4	0.5	**4.0**	0.1	0.0	**1.9**
Hard Syntax	54.5	24.2	**31.7**	18.7	6.1	6.2	**18.1**	-0.1	2.0	**11.9**
Gender	13.6	**22.1**	5.6	6.1	2.4	2.3	**5.0**	0.2	0.3	**4.7**
Temporal	47.8	-0.3	**-0.0**	-0.7	-0.4	**-0.3**	-0.6	-0.4	**0.4**	0.0
Average	40.2	**11.5**	10.8	8.3	2.3	**6.1**	5.6	-0.0	1.0	**4.1**

Table 6.3: Comparison of success rate improvements on model editing with canonical examples at three degradation balls for full finetuning, LoRA, and the sense finetuned Backpack ensemble for GPT-J. For the most conservative degradation ball, our Backpack methods outperforms the other methods.

language model (ours will be a Backpack), and $p_{\text{bp}}^{\text{ft}}$ be a language model finetuned on canonical examples. Intuitively, we want to impart the adaptations of the canonical example finetuning to a larger language model p_{large}. We do so by the following:

$$\log p_{\text{large}}^{\text{ft}} \propto \beta(\log p_{\text{bp}}^{\text{ft}} - \log p_{\text{bp}}^{\text{pre}}) + \log p_{\text{large}}^{\text{pre}}. \tag{6.11}$$

Intuitively, since the pretrained and finetuned Backpacks are within ϵ loss of each other, adding their difference of logits should only rarely make large changes to p_{large}.[7] This simple heuristic recently used in the setting of approximating finetuning large models by finetuning small models, by Mitchell et al. (2024).

Experiments & Results We use the GPT-J-6B model (Wang and Komatsuzaki, 2021), comparing full finetuning and LoRA finetuning to our proposed ensemble. We choose GPT-J since it uses the same tokenization as our Backpack. We do no further finetuning of the GPT-J model in the ensemble.[8] We run a 10-point random hyperparameter sweep on the validation set for the GPT-J finetuning methods.

Generalization results are in Table 6.3, and hard negatives results in Table C.4. We find that for the most strict degradation ball $B_{10^{-5}}$, our Backpack ensemble even substantially outperforms both finetuning methods for GPT-J in generalization, at no cost in hard negative performance. For the less strict degradation balls, our ensemble performs slightly worse than the other methods. This result is evidence that the Backpack with sense tuning is *more adaptable* than the 35x-larger GPT-J, and with our ensemble, we can impart the benefits of these adaptations to the larger model.

[7]We run a coarse search (in increments of 0.1) for a value of β as close to 1 as possible while ensuring the resulting model is in the correct degradation ball.

[8]Running both Backpacks takes only marginally more compute than running one (see Appendix C.1).

6.7.1 Visualizing Backpack improvements

To provide intuition for how sense finetuning updates a model, we provide two examples in Figure 6.5. The first canonical example is *The capital of Nauru is Yaren*. Because of their greater importance to the canonical example than to general text (Eqn 6.9), sense vectors of the subword `aur` in *Nauru* are chosen for finetuning. The result of finetuning is to increase the score of the subwords of Yaren, `Y` and `aren`, under the sense vector—this score is not dependent on context, and contributes additively to the model predictions with weight α. Thus, when the network chooses to look at the finetuned senses, it will **always** score the corresponding words more highly relative to the pretrained model. Thus, changing lexical associations are the most obvious uses for sense finetuning. In the canonical example *The sheriff said {he, she}*, sense vectors of *sheriff* are finetuned to score words like *her* more highly—but note that when an explicit pronoun is used in context, the model can still copy from the prior pronoun.

6.8 Discussion & Conclusion

In this chapter, we presented *model editing with canonical examples*, a problem setting that centers learning from a single example, evaluating out-of-distribution, and strictly limiting deviation from the original model. We've found that simple finetuning methods like LoRA can improve models somewhat with canonical examples while keeping the model's loss within a factor of $1 + 10^{-4}$. However, it is difficult to precisely edit models, especially since only string supervision is provided, as shown by the decrease in performance of MEMIT compared to its performance when it receives stronger supervision. We've shown that the Backpack's sense vectors provide a useful method for model editing with canonical examples, even for improving the 35x larger GPT-J model more than finetuning GPT-J itself in one setting. We hope that the setting of model editing with canonical examples will help spur research in understanding and robust improvement of LLMs.

The architecture of a neural model has implications not just for its computational efficiency and inductive bias, but also for the kinds of fixes we can make to it after it's trained. The Backpack and its lexically-defined sense vectors allow for precise edits of lexical selections. In exploring new model architectures, we suggest directly designing in components corresponding to the kinds of fixes we want to be able to make. While it's costly to train new models with new architectures, we can leverage small, adaptable models to fix monolithic large models, like we've shown here with GPT-J.

Chapter 7

Conclusion

In the last six years, language models have become artefacts of enough complexity, capability, and impact, that they deserve both fundamental exploratory study and preemtive design towards our ability to effectively control and fix them. These two directions, understanding *through discovery* and understanding *by design*, draw on very different notions of value in engineering science, and I believe both are crucial for long-term progress. Long-term understanding research, as I argue in the introduction, is useful to the extent to which it influences your intuitions as a researcher, and pushes you in different directions in each of the future research decisions you make.

In Part 1 of this thesis, corresponding to the papers Hewitt and Manning (2019), Hewitt and Liang (2019), and Hewitt et al. (2021), my coauthors and I grappled with questions surrounding how we find concepts we know about in models, which turned out to be a rich methodological question. When thinking back on this line of work as a whole, I think we (and others) did show that language models build representations that *go much of the way* from the input to the high-level linguistic concepts that we were searching for. Whether this is the same thing as the model approximately learning those concepts is up for debate, but I think it's largely a question of what language best aligns with your intuitions. To my intuition, yes, having constructed representations that make properties easy to predict *is* approximate knowledge of those properties. But as I state in the introduction, this is having knowledge in the sense that a library has—or contains—knowledge, not in the sense that an agent operating in the world has knowledge.

In Part 2 of this thesis, the chapters corresponding to the papers Hewitt et al. (2023) and Hewitt et al. (2024), my coauthors and I attempted to design language systems that are intended to be understood. Jointly with this goal, we tried to ensure that the resulting systems would scale in quality and usability with increasing computation and parameters. Either capable systems or understandable systems are hard goals separately, but together the goals may seem more difficult. In one sense this is true, but in another sense, as systems become stronger, they can learn hooks that we can use to understand them. This was, to me, the key takeaway of the Backpack. The failures of this work, however, were that the resulting control or fixes we could perform in the models, in the Backpack in particular, were simply too low-level to be of sufficient interest to the wider community. So, while I like the ideas of the Backpack and its properties have influenced my thoughts since then, I find it unlikely a year and a half onward from publication that it will gain popularity. This is okay to me, but also a lesson for future attempts at bridging understanding and control.

I see the future of understanding language models as pursuing intuitive understanding, and then grounding and verifying that understanding in concrete control and improvements to existing models. Right now, the best recipes we have for control of language models are largely unmotivated from an understanding perspective (though there are exceptions, e.g., Wu et al. (2024).) I hope dedicated work in understanding and evaluation changes this, and I encourage scientists interested in understanding models to not shy away from competing, in some sense, with methods that are unmotivated from an understanding perspective. However, I also hope to read surprising blue sky papers that discover wild properties of networks with no obvious immediate application. Let us discover things and ponder them, and eventually put them to good use. Models are increasingly deployed though and increasingly impactful in the world, so I believe fixing them is a priority.

If you've made it this far, it's been quite a journey. If you're a scientist, engineer, PhD student, master's student, undergraduate student, wishing you were a student, wishing you weren't a student—I encourage you to work on problems that you feel will lead you to deeper understanding of interesting phenomena. This might mean building systems, or breaking them, or studying their use, or seeing what makes them tick, or any number of other things. One interesting thing in the last six years of machine learning-driven language system research is that things have begun to *work* much more than they ever had in the past. We've developed recipes for building capable systems that are pretty reliable in building artefacts that are useful for a range of things (at least, I've found them useful!) When things don't work, there's a natural tendency for the entropy of the distribution of research directions to increase. Everyone tries different things, because none of them work. When things start to work, there's a natural decrease in entropy. Many people feel the excitement and want to iterate on the methods that work to make them work even better! However, at its extremes, this can lead to a research monoculture that I think holds back the field in the long term. By pursuing directions that you feel will give you deep understanding of a phenomenon of interest, I think you'll naturally find yourself doing work you enjoy and that meets some kind of middle ground between pursuing things that work and things that might not work yet but are just fascinating.

Bibliography

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*.

Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. 2021. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34:4699–4711.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Boulder, Colorado. Association for Computational Linguistics.

Afra Feyza Akyürek, Eric Pan, Garry Kuwanto, and Derry Tanti Wijaya. 2023. DUnE: Dataset for unified editing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Dimosthenis Antypas, Asahi Ushio, Jose Camacho-Collados, Vitor Silva, Leonardo Neves, and Francesco Barbieri. 2022. Twitter topic classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3386–3400, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Peter L Bartlett and Shahar Mendelson. 2001. Rademacher and gaussian complexities: Risk bounds and structural results. In *International Conference on Computational Learning Theory*, pages 224–240. Springer.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. 2020a. Rewriting a deep generative model. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 351–369. Springer.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. 2020b. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078.

Yonatan Belinkov. 2021. Probing classifiers: Promises, shortcomings, and alternatives. *CoRR*, abs/2102.12452.

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. Association for Computational Linguistics.

Yonatan Belinkov and James Glass. 2017. Analyzing hidden representations in end-to-end automatic speech recognition systems. In *Advances in Neural Information Processing Systems*, pages 2441–2451.

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

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. 2024. Impossibility theorems for feature attribution. *Proceedings of the National Academy of Sciences of the United States of America*, 121(2):e2304406120.

Arianna Bisazza and Clara Tump. 2018. The lazy encoder: A fine-grained analysis of the role of morphology in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2871–2876.

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. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. *Advances in neural information processing systems*, 29.

Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Steven Cao, Victor Sanh, and Alexander Rush. 2021. Low-complexity probing via finding subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 960–966, Online. Association for Computational Linguistics.

Chun-Hao Chang, Rich Caruana, and Anna Goldenberg. 2022. NODE-GAM: Neural generalized additive model for interpretable deep learning. In *International Conference on Learning Representations*.

Boli Chen, Yao Fu, Guangwei Xu, Pengjun Xie, Chuanqi Tan, Mosha Chen, and Liping Jing. 2021. Probing BERT in hyperbolic spaces. In *International Conference on Learning Representations*.

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding universal grammatical relations in multilingual BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577, Online. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton.

Gabriella Chronis and Katrin Erk. 2020. When is a bishop not like a rook? When it's like a rabbi! Multi-prototype BERT embeddings for estimating semantic relationships. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 227–244.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 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. Association for Computational Linguistics.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit optimizers via block-wise quantization. *9th International Conference on Learning Representations, ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics.

Michel Marie Deza and Monique Laurent. 2009. *Geometry of cuts and metrics*, volume 15. Springer.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Abhimanyu Dubey, Filip Radenovic, and Dhruv Mahajan. 2022. Scalable interpretability via polynomials. In *Advances in Neural Information Processing Systems*.

Tiwalayo Eisape, Vineet Gangireddy, Roger P. Levy, and Yoon Kim. 2022. Probing for incremental parse states in autoregressive language models. In *Findings of EMNLP 2022*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

H.C. Ellis. 1965. *The Transfer of Learning*. Critical issues in psychology series. Macmillan.

Zied Elloumi, Laurent Besacier, Olivier Galibert, and Benjamin Lecouteux. 2018. Analyzing learned representations of a deep ASR performance prediction model. *arXiv preprint arXiv:1808.08573*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1790–1801.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 134–139.

Li Fei-Fei, R. Fergus, and P. Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Michael Fink. 2004. Object classification from a single example utilizing class relevance metrics. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

J. R. Firth. 1935. The technique of semantics. *Transactions of the Philological Society*, 34(1):36–73.

J. R. Firth. 1957. Applications of general linguistics. *Transactions of the Philological Society*, 56(1):1–14.

Richard Futrell, Ethan Wilcox, Takashi Morita, and Roger Levy. 2018. RNNs as psycholinguistic subjects: Syntactic state and grammatical dependency. *arXiv preprint arXiv:1809.01329*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*.

Robert Geirhos, Roland S. Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. 2024. Don't trust your eyes: on the (un)reliability of feature visualizations. In *Forty-first International Conference on Machine Learning*.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182, Austin, Texas. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Yoav Goldberg. 2017. Neural network methods for natural language processing. `https://nbviewer.org/gist/yoavg/d76121dfde2618422139`. Accessed: 2024-08-07.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*.

Albert Gu, Karan Goel, and Christopher Re. 2021. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

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)*, volume 1, pages 1195–1205.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 12–21.

Prakhar Gupta and Martin Jaggi. 2021. Obtaining better static word embeddings using contextual embedding models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5241–5253.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? Surprising differences in causality-based localization vs. knowledge editing in language models.

Trevor Hastie and Robert Tibshirani. 1986. Generalized additive models. *Statistical Science*, 1(3):297–318.

Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023. Measuring and manipulating knowledge representations in language models. *arXiv preprint arXiv:2304.00740*.

John Hertz, Anders Krogh, and Richard G Palmer. 1991. Introduction to the theory of neural computation.

John Hewitt, Sarah Chen, Lanruo Lora Xie, Edward Adams, Percy Liang, and Christopher D Manning. 2024. Model editing with canonical examples. *arXiv preprint arXiv:2402.06155*.

John Hewitt, Kawin Ethayarajh, Percy Liang, and Christopher D Manning. 2021. Conditional probing: measuring usable information beyond a baseline. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1639.

John Hewitt and Percy Liang. 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.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. Association for Computational Linguistics.

John Hewitt, Christopher D Manning, and Percy Liang. 2022. Truncation sampling as language model desmoothing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3414–3427.

John Hewitt, John Thickstun, Christopher D. Manning, and Percy Liang. 2023. Backpack language models. In *Association for Computational Linguistics (ACL)*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

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

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. 2001. Learning to learn using gradient descent. In *Artificial Neural Networks—ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*, pages 87–94. Springer.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*.

Yifan Hou and Mrinmaya Sachan. 2021. Bird's eye: Probing for linguistic graph structures with a simple information-theoretic approach. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1844–1859, Online. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Wilhelm von Humboldt. 1836. Über die verschiedenheit des menschlichen sprachbaues und ihren einfluss auf die geistige entwickelung des menschengeschlechts.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Anna A. Ivanova, John Hewitt, and Noga Zaslavsky. 2021. Probing artificial neural networks: insights from neuroscience. In *Proceedings of the Brain2AI Workshop at the Ninth International Conference on Learning Representations*.

Anna A Ivanova, Martin Schrimpf, Stefano Anzellotti, Noga Zaslavsky, Evelina Fedorenko, and Leyla Isik. 2022. Beyond linear regression: mapping models in cognitive neuroscience should align with research goals. *arXiv preprint arXiv:2208.10668*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall. N-gram Language Models.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Najoung Kim, Roma Patel, Adam Poliak, Alex Wang, Patrick Xia, R Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. 2019. Probing what different nlp tasks teach machines about function word comprehension. *arXiv preprint arXiv:1904.11544*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53, New Orleans, Louisiana. Association for Computational Linguistics.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR.

Arne Köhn. 2015. What's in an embedding? Analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888.

Yair Lakretz, Germán Kruszewski, Théo 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.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Tomasz Limisiewicz and David Mareček. 2021. Introducing orthogonal constraint in structural probes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 428–442, Online. Association for Computational Linguistics.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the First Workshop on Evaluating Vector Space Representations for NLP*. Association for Computational Linguistics.

Tal Linzen. 2018. What can linguistics and deep learning contribute to each other? *arXiv preprint arXiv:1809.04179*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Tal Linzen and Brian Leonard. 2018. Distinct patterns of syntactic agreement errors in recurrent networks and humans. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 692–697. Cognitive Science Society, Austin, TX.

Zachary C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. 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 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (Workshop Poster)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. 2021. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7721–7735. PMLR.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D Manning. 2024. An emulator for fine-tuning large language models using small language models. In *The Twelfth International Conference on Learning Representations*.

Richard Montague. 1970. Universal grammar. *Theoria*, 36(3):373–398.

Shikhar Murty, Christopher D Manning, Scott Lundberg, and Marco Tulio Ribeiro. 2022. Fixing model bugs with natural language patches. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11600–11613.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Benjamin Newman, Kai-Siang Ang, Julia Gong, and John Hewitt. 2021. Refining targeted syntactic evaluation of language models. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. 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 4227–4237, Hong Kong, China. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020a. Pareto probing: Trading off accuracy for complexity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3138–3153, Online. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 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.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Association for Computational Linguistics (ACL)*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Investigating language universal and specific properties in word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1478–1488.

Filip Radenovic, Abhimanyu Dubey, and Dhruv Mahajan. 2022. Neural basis models for interpretability. In *Advances in Neural Information Processing Systems*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Open AI Technical Report*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. *Advances in neural information processing systems*, 32.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. The (Too Many) Problems of Analogical Reasoning with Word Vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 135–148.

Douglas LT Rohde, Laura M Gonnerman, and David C Plaut. 2005. An improved model of semantic similarity based on lexical co-occurrence.

Rudolf Rosa, Tomáš Musil, and David Mareček. 2020. Measuring memorization effect in word-level neural networks probing. In *Text, Speech, and Dialogue*, pages 180–188, Cham. Springer International Publishing.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633.

Gözde Gül Şahin, Clara Vania, Ilia Kuznetsov, and Iryna Gurevych. 2019. Linspector: Multilingual probing tasks for word representations. *arXiv preprint arXiv:1903.09442*.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with svcca. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. Association for Computational Linguistics.

Marten van Schijndel and Tal Linzen. 2018. Modeling garden path effects without explicit hierarchical syntax. In Tim Rogers, Marina Rau, Jerry Zhu, and Chuck Kalish, editors, *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 2600–2605. Cognitive Science Society, Austin, TX.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Language Resources and Evaluation (LREC)*.

H. Schütze. 1992. Dimensions of meaning. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, Supercomputing '92, page 787–796, Washington, DC, USA. IEEE Computer Society Press.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Naomi Tachikawa Shapiro, Amandalynne Paullada, and Shane Steinert-Threlkeld. 2021. A multilabel approach to morphosyntactic probing. *arXiv preprint arXiv:2104.08464*.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *International Conference on Machine Learning*.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language models via meta learning. *arXiv preprint arXiv:2311.04661*.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? A targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Peter D Turney. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Leslie G Valiant. 1984. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

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

Elena Voita and Ivan Titov. 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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. *OntoNotes release 5*. LDC2013T19.

Jennifer C. White, Tiago Pimentel, Naomi Saphra, and Ryan Cotterell. 2021. A non-linear structural probe. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 132–138, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. 2022. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971.

Zhengxuan* Wu, Aryaman* Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2024. Reft: Representation finetuning for language models.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. 2020. A theory of usable information under computational constraints. In *International Conference on Learning Representations*.

Zebin Yang, Aijun Zhang, and Agus Sudjianto. 2021. GAMI-Net: An explainable neural network based on generalized additive models with structured interactions. *Pattern Recognition*, 120:108192.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*.

Kelly W Zhang and Samuel R Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint arXiv:1809.10040*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

Appendix A

Formal Results on Multivariate $\mathcal{V}$-Information.

A.1 Multivariable $\mathcal{V}$-information

In this section we introduce Multivariable $\mathcal{V}$-information. $\mathcal{V}$-information as introduced by Xu et al. (2020) was defined in terms of a single predictive variable X, and is unwieldy to extend to multiple variables due to its use of a "null" input outside the sample space of X (Section A.4.3).[1] Our multivariable V-information removes the use of null variables and naturally captures the multivariable case. Consider an agent attempting to predict $Y \in \mathcal{Y}$ from some information sources $X_1, \dots, X_n$, where $X_i \in \mathcal{X}_i$. Let $\mathcal{P}(\mathcal{Y})$ be the set of all probability distributions over Y.

At a given time, the agent may only have access to a subset of the information sources. Let the known set $C \in \mathcal{C}$ and unknown set $\bar{C} \in \bar{\mathcal{C}}$ be a binary partition of $X_1, \dots, X_n$. Though the agent isn't given the true value of $\bar{C}$ when predicting Y, it is instead provided with a constant value $\bar{a} \in \bar{\mathcal{C}}$, which does not vary with Y.[2]

We first specify constraints on the set of functions that the agent has at its disposal for predicting Y from X:

Definition 1 (Multivariable Predictive Family)**.** *Let $\Omega = \{f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{P}(\mathcal{Y})\}$. We say that $\mathcal{V} \subseteq \Omega$ is a predictive family if, for any partition of $\mathcal{X}_1, \dots, \mathcal{X}_n$ into $\mathcal{C}, \bar{\mathcal{C}}$, we have*

$$\begin{aligned} &\forall f, x_1, \dots, x_n, \in \mathcal{V} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_n, \\ &\exists f' \in \mathcal{V} : \forall \bar{c}' \in \bar{\mathcal{C}},\ f(c, \bar{c}) = f'(c, \bar{c}'), \end{aligned} \tag{A.1}$$

where we overload $f(c, \bar{c})$ to equal $f(x_1, \dots, x_n)$ for the values of $x_1, \dots, x_n$ specified by $c, \bar{c}$.

[1] In particular, the null input encodes *not* knowing the value of X; technical conditions in the definition of $\mathcal{V}$-information as to the behavior of this null value increase in number exponentially with the number of predictive variables.

[2] The exact value of $\bar{a}$ will not matter, as a result of Definition 1.

Intuitively, the constraint on $\mathcal{V}$ states that for any binary partition of the $X_1, \ldots, X_n$ into known and unknown sets, if a function is expressible given some constant assignment to the unknown variables, the same function is expressible if the unknown variables are allowed to vary arbitrarily. Intuitively, this means one can assign zero weight to those variables, so their values don't matter. This constraint, which we refer to as *multivariable optional ignorance* in reference to Xu et al. (2020), will be used to ensure non-negativity of information; when some X_ℓ is moved from $\bar{C}$ to C as a new predictive variable for the agent to use, optional ignorance ensures the agent can still act as if that variable were held constant.

Example 1. *Let $X_1, \ldots, X_n \in \mathbb{R}^{d_1}, \ldots, \mathbb{R}^{d_n}$ and $Y \in \mathcal{Y}$ be random variables. Let Ω be defined as in Definition 1. Then $V = \{f : f(x_1, \ldots, x_n) = \textit{softmax}(W_2\ \sigma(W_1[x_1; \cdots; x_n] + b) + b)\}$, the set of 1-layer multi-layer perceptrons, is a predictive family. Ignorance of some x_i can be achieved by setting the corresponding rows of W_1 to zero.*

Given the predictive family of functions the agent has access to, we define the multivariable $\mathcal{V}$-information analogue of entropy:

Definition 2 (Multivariable Predictive $\mathcal{V}$-entropy)**.** *Let $X_1, \ldots, X_n \in \mathcal{X}_1, \ldots, \mathcal{X}_n$. Let $C \in \mathcal{C}$ and $\bar{C} \in \bar{\mathcal{C}}$ form a binary partition of $X_1, \ldots, X_n$. Let $\bar{a} \in \bar{\mathcal{C}}$. Then the $\mathcal{V}$-entropy of Y conditioned on C is defined as*

$$H_{\mathcal{V}}(Y|C) = \inf_{f \in V} \mathbb{E}_{c,y}\big[-\log f(c, \bar{a})[y]\big]. \tag{A.2}$$

Note that $\bar{a}$ does not vary with y; thus it is 'informationless'. The notation $f(c, \bar{a})$ takes the known value of $C \subseteq \{X_1, \ldots, X_n\}$, and the constant value $\bar{a}$, and produces a distribution over $\mathcal{Y}$, and $f(c, \bar{a})[y]$ evaluates the density at y.

If we let $V = \Omega$, the set of all functions from the $\mathcal{X}_i$ to distributions over $\mathcal{Y}$, then $\mathcal{V}$-entropy becomes exactly Shannon entropy (Xu et al., 2020). And just like for Shannon information, the multivariable $\mathcal{V}$-information from some variable X_ℓ to Y is defined as the reduction in entropy when its value becomes known. In our notation, this means some X_ℓ is moving from $\bar{C}$ (the unknown variables) to C (the known variables), so this definition encompasses the notion of *conditional* mutual information if C is non-empty to start.

Definition 3 (Multivariable $\mathcal{V}$-information)**.** *Let $X_1, \ldots, X_n \in \mathcal{X}_1, \ldots, \mathcal{X}_n$ and $Y \in \mathcal{Y}$ be random variables. Let $\mathcal{V}$ be a multivariable predictive family. Then the conditional multivariable $\mathcal{V}$-information from X_ℓ to Y, where $\ell \in \{1, \ldots, n\}$, conditioned on prior knowledge of $C \subset \{X_1, \ldots, X_n\}$, is defined as*

$$I_{\mathcal{V}}(X_\ell \rightarrow Y|C) = H_{\mathcal{V}}(Y|C) - H_{\mathcal{V}}(Y|C \cup \{X_\ell\}) \tag{A.3}$$

A.1.1 Properties of multivariable $\mathcal{V}$-information

The crucial property of multivariable $\mathcal{V}$-information as a descriptor of probing is that it can be *constructed* through computation. In the example of the agent attempting to predict the sentiment (Y) of an encrypted message (X), if the agent has $\mathcal{V}$ equal to the set of linear functions, then $I_{\mathcal{V}}(X \rightarrow Y)$ is small[3]. A function ϕ that decrypts the message constructs $\mathcal{V}$-information about Y, since $I_{\mathcal{V}}(\phi(X) \rightarrow Y)$ is larger. In probing, ϕ is interpreted to be the contextual representation learner, which is interpreted as *constructing* $\mathcal{V}$-information about linguistic properties.

$\mathcal{V}$-information also has some desirable elementary properties, including preserving some of the properties of mutual information, like non-negativity. (Knowing some X_ℓ should not reduce the agent's ability to predict Y).

Proposition 1. *Let $X_1, \ldots, X_n \in \mathcal{X}_1, \ldots, \mathcal{X}_n$ and $Y \in \mathcal{Y}$ be random variables, and $\mathcal{V}$ and $\mathcal{U}$ be predictive families. Let $C, \bar{C}$ be a binary partition of $X_1, \ldots, X_n$.*

1. ***Independence** If Y, C are jointly independent of X_ℓ, then $I_{\mathcal{V}}(X_\ell \rightarrow Y|C) = 0$.*
2. ***Monotonicity** If $\mathcal{U} \subseteq \mathcal{V}$, then $H_{\mathcal{V}}(Y|C) \leq H_{\mathcal{U}}(Y|C)$.*
3. ***Non-negativity** $I_{\mathcal{V}}(X_\ell \rightarrow Y|C) \geq 0$.*

A.2 Probing as Multivariable $\mathcal{V}$-information Estimation

We've described the $\mathcal{V}$-information framework, and discussed how it captures the intuition that usable information about linguistic properties is constructed through contextualization. In this section, we demonstrate how a small step from existing probing methodology leads to probing estimating $\mathcal{V}$-information quantities.

A.2.1 Estimating $\mathcal{V}$-entropy

In probing, gradient descent is used to pick the function in $\mathcal{V}$ that minimizes the cross-entropy loss,

$$\frac{1}{\mathcal{D}_{\text{tr}}} \sum_{x,y \in \mathcal{D}_{\text{tr}}} -\log p(y|\phi_i(x); \theta), \tag{A.4}$$

where θ are the trainable parameters of functions in $\mathcal{V}$. Recalling the definition of $\mathcal{V}$-entropy, this minimization performed through gradient descent is approximating the $\inf$ over $\mathcal{V}$, since $-\log p(y|x;\theta)$ is equal to $-\log f_\theta(x)[y]$. To summarize, this states that the supervision used in probe training can be interpreted as approximating the $\inf$ in the definition of $\mathcal{V}$-entropy. In traditional probing, the performance of the probe is measured on the test set $\mathcal{D}_{\text{te}}$ using the traditional metric of the task, like accuracy of F_1 score. In $\mathcal{V}$-information probing, we use $\mathcal{D}_{\text{te}}$ to approximate the expectation in the definition of $\mathcal{V}$-entropy. Thus, the performance of a single probe on representation R, where the performance metric is cross-entropy loss, is an estimate of $H_{\mathcal{V}}(Y|R)$. This brings us to our framing of a probing experiment as estimating a $\mathcal{V}$-information quantity.

[3]Where $I_{\mathcal{V}}(X \rightarrow Y)$ is defined to be $I_{\mathcal{V}}(X \rightarrow Y|\{\})$

A.2.2 Baselined probing

Let baselined probing be defined as in the main paper. Then if the performance metric is defined as the negative cross-entropy loss, we have that Perf(B) estimates $-H_{\mathcal{V}}(Y|B)$, Perf$(\phi(X))$ estimates $-H_{\mathcal{V}}(Y|\phi(X))$, and so baselined probing performance is an estimate of

$$\begin{aligned}&H_{\mathcal{V}}(Y|\{B\}) - H_{\mathcal{V}}(Y|\{\phi_i(X)\})\\&= I_{\mathcal{V}}(\phi_i(X) \to Y) - I_{\mathcal{V}}(B \to Y)\end{aligned} \tag{A.5}$$

A.2.3 Conditional probing

Let conditional probing be defined as in the main paper. Then if the performance metric is defined as the negative cross-entropy loss, we have that Perf$([B; \mathbf{0}])$ estimates $-H_{\mathcal{V}}(Y|B)$, Perf$([B; \phi(X)])$ estimates $-H_{\mathcal{V}}(Y|B, \phi(X))$, and so conditional probing performance is an estimate of

$$\begin{aligned}&H_{\mathcal{V}}(Y|\{B\}) - H_{\mathcal{V}}(Y|\{B, \phi_i(X)\})\\&= I_{\mathcal{V}}(\phi_i(X) \to Y|B)\end{aligned} \tag{A.6}$$

The first is estimated with a probe just on B—under the definition of predictive family, this means providing the agent with the real values of the baseline, and some constant value like the zero vector instead of $\phi_i(X)$. That is, holding $\bar{a} \in \phi_i(\mathcal{X})_i$ constant and sampling $b, y \sim B, Y$, the probability assigned to y is $f(b, \bar{a})[y]$ for $f \in \mathcal{V}$. The second term is estimate with a probe on both B and $\phi_i(X)$. So, sampling $b, x, y \sim B, X, Y$, the probability assigned to y is $f(b, \phi_i(x))[y]$ for $f \in \mathcal{V}$. Intuitively, conditional probing measures the *new* information in $\phi_i(X)$ because in both probes, the agent has access to B, so no benefit is gained from $\phi_i(X)$ supplying the same information.

A.3 Proof of Proposition 1

Monotonicity *If $\mathcal{U} \subseteq \mathcal{V}$, then $H_{\mathcal{V}}(Y|C) \leq H_{\mathcal{U}}(Y|C)$.* Proof:

$$\begin{aligned}H_{\mathcal{U}}(Y|C) &= \inf_{f\in\mathcal{U}} \mathbb{E}_{c,y}\left[-\log f[c,\bar{a}](y)\right]\\&\geq \inf_{f\in\mathcal{V}} \mathbb{E}_{c,y}\left[-\log f[c,\bar{a}](y)\right]\\&= H_{\mathcal{V}}(Y|C)\end{aligned} \tag{A.7}$$

This holds because we are taking the infimum over $\mathcal{V}$ such that if $f \in \mathcal{U}$ then $f \in \mathcal{V}$.

Non-Negativity $I_{\mathcal{V}}(X_\ell \to Y|C) \geq 0$. *Where $\mathcal{V}_{\bar{C}} \subset \mathcal{V}$ is the subset of functions that satisfies $f[c, \bar{c}] = f[c, \bar{c}'] \;\forall\, \bar{c}' \in \bar{\mathcal{C}}$, and $\bar{a}, \bar{a}_{/\ell}$ denote the constant values of the unknown set with and without X_ℓ, the proof is as follows:*

$$\begin{aligned}
H_{\mathcal{V}}(Y|C) &= \inf_{f\in\mathcal{V}} \mathbb{E}_{c,x_\ell,y}\left[-\log f[c,\bar{a}](y)\right] \\
&= \inf_{f\in\mathcal{V}_{\bar{C}}} \mathbb{E}_{c,x_\ell,y}\left[-\log f[c,x_\ell,\bar{a}_{/\ell}](y)\right] \\
&\geq \inf_{f\in\mathcal{V}} \mathbb{E}_{c,x_\ell,y}\left[-\log f[c,x_\ell,\bar{a}_{/\ell}](y)\right] \\
&= H_{\mathcal{V}}(Y|C\cup\{X_\ell\})
\end{aligned} \tag{A.8}$$

By definition, $I_{\mathcal{V}}(X_\ell \to Y|C) = H_{\mathcal{V}}(Y|C) - H_{\mathcal{V}}(Y|C\cup\{X_\ell\}) \geq 0$.

Independence *If Y, C are jointly independent of X_ℓ, then $I_{\mathcal{V}}(X \to Y|C) = 0$. Proof:*

$$\begin{aligned}
&H_{\mathcal{V}}(Y|C\cup\{X_\ell\}) \\
&= \inf_{f\in\mathcal{V}} \mathbb{E}_{c,x_\ell,y}\left[-\log f[c,x_\ell,\bar{a}_{/\ell}](y)\right] \\
&= \inf_{f\in\mathcal{V}} \mathbb{E}_{x_\ell}\mathbb{E}_{c,y}\left[-\log f[c,x_\ell,\bar{a}_{/\ell}](y)\right] \\
&\geq \mathbb{E}_{x_\ell}\left[\inf_{f\in\mathcal{V}} \mathbb{E}_{c,y}\left[-\log f[c,x_\ell,\bar{a}_{/\ell}](y)\right]\right] \\
&= \mathbb{E}_{x_\ell}\left[\inf_{f\in\mathcal{V}_{\bar{C}}} \mathbb{E}_{c,y}\left[-\log f[c,x_\ell,\bar{a}_{/\ell}](y)\right]\right] \\
&= \inf_{f\in\mathcal{V}_{\bar{C}}} \mathbb{E}_{c,y}\left[-\log f[c,\bar{a}](y)\right] \\
&\geq \inf_{f\in\mathcal{V}} \mathbb{E}_{c,y}\left[-\log f[c,\bar{a}](y)\right] \\
&= H_{\mathcal{V}}(Y|C)
\end{aligned} \tag{A.9}$$

In the second line, we break down the expectation based on conditional independence. Then we apply Jensen's inequality and optional ignorance to remove the expectation w.r.t. x. Since $\mathcal{V}_{\bar{C}} \subset \mathcal{V}$, the former's infimum is at least as large as the latter's. Then

$$I_{\mathcal{V}}(X_\ell \to Y|C) = H_{\mathcal{V}}(C) - H_{\mathcal{V}}(Y|C\cup\{X_\ell\}) \leq 0$$

Combined with non-negativity (i.e., $I_{\mathcal{V}}(X_\ell \to Y|C) > 0$) we have inequality in both directions, so $I_{\mathcal{V}}(X_\ell \to Y|C) = 0$.

A.4 Equivalence of Xu et al. (2020) and our $\mathcal{V}$-information

In order to define conditional probing, we needed a theory of $\mathcal{V}$-information that considered arbitrarily many predictive variables $\mathcal{X}_1, \ldots, \mathcal{X}_n$. $\mathcal{V}$-information as presented by Xu et al. (2020) considers only a single predictive variable $\mathcal{X}$. It becomes extremely cumbersome, due to the use of null variables in their presentation, to expand this to more, let alone arbitrarily many variables. So, we redefined and extended $\mathcal{V}$-information to more naturally capture the case with an arbitrary (finite) number of variables. In this section, we show that, in the single predictive variable case considered by Xu et al. (2020), our $\mathcal{V}$-information definition is equivalent to theirs. For the sake of this section, we'll call the $\mathcal{V}$-information of Xu et al. (2020) Xu-$\mathcal{V}$-information, and ours $\mathcal{V}$-information.

In particular, we show that there is a transformation from any predictive family of Xu-$\mathcal{V}$-information to predictive family for $\mathcal{V}$-information under which predictive $\mathcal{V}$-entropies are the same (and the same in the opposite direction.)

A.4.1 From Xu et al. (2020) to ours

We recreate the definition of predictive family from Xu et al. (2020) here:

Definition 4 (Xu predictive family)**.** *Let $\Upsilon = \{f : \mathcal{X} \cup \{\varnothing\} \to \mathcal{P}(\mathcal{Y})\}$. We say that $\mathcal{U} \subseteq \Upsilon$ is a Xu predictive family if it satisfies*

$$\forall f \in \mathcal{U}, \forall P \in \mathit{range}(f), \exists f' \in \mathcal{U}, s.t. \tag{A.10}$$

$$\forall x \in \mathcal{X}, f[x] = P, f'[\varnothing] = P \tag{A.11}$$

Now, we construct one of our predictive families from the Xu predictive family. Let $\mathcal{U} \subseteq \Upsilon$ be a Xu predictive family. We now construct a predictive family under our framework, $\mathcal{V} \subseteq \Omega$. For each $f \in \mathcal{U}$, $f : \mathcal{X} \cup \{\varnothing\} \to \mathcal{P}(\mathcal{Y})$, construct the following two functions: first, g, which recreates the behavior of f on the domain of $\mathcal{X}$:

$$g : \mathcal{X} \to \mathcal{P}(\mathcal{Y}) \tag{A.12}$$

$$g : x \mapsto f(x) \tag{A.13}$$

and second, g', which recreates the behavior of f on $\varnothing$, given any input from $\mathcal{X}$:

$$g' : \mathcal{X} \to \mathcal{P}(\mathcal{Y}) \tag{A.14}$$

$$g' : x \mapsto f(\varnothing) \tag{A.15}$$

Then we define our predictive family as the union of g, g' for all $f \in \mathcal{V}$:

$$\mathcal{V} = \bigcup_{f \in \mathcal{U}} \{g, g'\}, \tag{A.16}$$

where $\mathcal{V} \subseteq \Omega$ and $\Omega = \{f : \mathcal{X} \to \mathcal{P}(\mathcal{Y})\}$. Note from this construction that we've eliminated the presence of the null variable from the definition of predictive family.

We now show that $\mathcal{V}$, as defined in the construction above, is in fact a predictive family under our definition.

Under our definition, there are two cases: either $\mathcal{X} \in C$ or $\mathcal{X} \in \bar{C}$. If $\mathcal{X} \in C$, then for all $f, x \in \mathcal{V} \times \mathcal{X}$, if we take any $f' \in \mathcal{V}$ (which is non-empty), then there is no $\bar{c}' \in \bar{C}$, so vacuously the condition holds. If $\mathcal{X} \in \bar{C}$, then for all $f, x \in \mathcal{V}$, we have that f was either g or g' for some function $h \in \mathcal{U}$ in the construction of $\mathcal{V}$ (because all functions in $\mathcal{V}$ were part of some pair g, g'.) Then we take $f' = g'$, and have that for all $\bar{c}' \in \bar{C}$, that is $x \in \mathcal{X}$, $f'(c, \bar{c}) = f'(c, \bar{c}') = g'(x) = h(\varnothing)$, satisfying the constraint.

Finally, we show that the predictive $\mathcal{V}$-entropies of $\mathcal{V}$ (under our definition) and $\mathcal{U}$ (under that of Xu et al. (2020)) are the same. Consider Xu-predictive entropies:

$$H_{\mathcal{U}}(Y|X) = \inf_{f \in \mathcal{U}} \mathbb{E}_{x,y}[-\log f[x](y)] \tag{A.17}$$

$$H_{\mathcal{U}}(Y|\varnothing) = \inf_{f \in \mathcal{U}} \mathbb{E}_{y}[-\log f[\varnothing](y)] \tag{A.18}$$

First we want to show $H_{\mathcal{U}}(Y|X) = H_{\mathcal{V}}(Y|X)$. Consider the $\inf$ in Equation A.17; the $f \in \mathcal{U}$ that achieves the $\inf$ corresponds to some $g \in \mathcal{V}$ by construction, and since $f(x) = g(x)$, we have that the value of the inf for $\mathcal{V}$ is at least as low as for $\mathcal{U}$. The same is true in the other direction; in our definition $H_{\mathcal{V}}(Y|\mathcal{X})$, the g that achieves the $\inf$ corresponds to some $f \in \mathcal{U}$ that produces the same probability distributions. So, $H_{\mathcal{U}}(Y|X) = H_{\mathcal{V}}(Y|X)$.

Now we want to show $H_{\mathcal{U}}(Y|\varnothing) = H_{\mathcal{V}}(Y)$. Now, consider the $\inf$ in Equation A.18. The $f \in \mathcal{U}$ that achieves the $\inf$ corresponds to some g, g' in the construction of $\mathcal{V}$; that g' takes any $x \in \mathcal{X}$ and produces $f[\varnothing]$; hence the value of the inf for $\mathcal{V}$ is at least as low as for $\mathcal{U}$. The same is true in the other direction. We have that $H_{\mathcal{V}}(Y) = \inf_{f \in \mathcal{V}} \mathbb{E}_y[-\log f(\bar{a})[y]]$ for any $\bar{a} \in \mathcal{X}$. Either a g or a g' from the construction of $\mathcal{V}$ achieves this $\inf$; if a g achieves it, then its corresponding g' emits the same probability distributions, so WLOG we'll assume it's a g'. We know that $g'(\bar{a}) = f(\varnothing)$ for all $\bar{a} \in \mathcal{X}$, so $H_{\mathcal{U}}(Y|\varnothing)$ is at most $H_{\mathcal{V}}(Y)$. So, $H_{\mathcal{U}}(Y|\varnothing) = H_{\mathcal{V}}(Y)$.

Since the V-entropies of the predictive family from Xu et al. (2020) and ours are the same, all the information quantities are the same. This shows that the predictive family we constructed in our theory is equivalent to the predictive family from Xu et al. (2020) that we started with.

A.4.2 From our $\mathcal{V}$-information to that of Xu et al. (2020)

Now we construct a predictive family $\mathcal{U}$ under the framework of Xu et al. (2020) from an arbitrary predictive family $\mathcal{V}$ under our framework. For each function $f \in \mathcal{V}$, we have from the definition that there exists $f' \in \mathcal{V}$ such that $\forall x \in \mathcal{X}$, $f'(x) = P$ for some $P \in \mathcal{P}(\mathcal{Y})$. We then define the function:

$$g : \mathcal{X} \cup \{\varnothing\} \to \mathcal{P}(\mathcal{Y}) \tag{A.19}$$

$$g(x) = \begin{cases} f(x) & x \in \mathcal{X} \\ f'(\bar{a}) & x = \varnothing \end{cases} \tag{A.20}$$

where $\bar{a} \in \mathcal{X}$ is an arbitrary element of $\mathcal{X}$, and the set of constant-valued functions

$$G = \{g' : g'(z) = P \mid P \in \text{range}(f)\}, \tag{A.21}$$

where $z \in \mathcal{X} \cup \{\varnothing\}$, and let

$$\mathcal{U} = \bigcup_{f \in \mathcal{V}} \{g\} \cup G \tag{A.22}$$

The set $\mathcal{U}$ is a predictive family under Xu-$\mathcal{V}$-information because for any $f \in \mathcal{U}$, f is either a g or a g' in our construction, and so optional ignorance is maintained by the set G that was either constructed for g or that g' was a part of. That is, from the construction, G contains a function for each element in the range of g (or g') that maps all $x \in \mathcal{X}$ as well as $\varnothing$ to that element, and $\mathcal{U}$ contains all elements in G.

Now we show that the predictive $\mathcal{V}$-entropies of $\mathcal{U}$ (from this construction) under Xu et al. (2020) are the same as for $\mathcal{V}$ under our framework.

First we want to show $H_{\mathcal{U}}(Y|X) = H_{\mathcal{V}}(Y|X)$. For the g that achieves the $\inf$ over $\mathcal{U}$ in Equation A.17, we have there exists $f \in \mathcal{V}$ such that $g(x) = f(x)$ given that $x \in \mathcal{X}$, so $H_{\mathcal{V}}(y|x) \leq H_{\mathcal{U}}(y|x)$ The same is true in the other direction; the $f \in \mathcal{V}$ that achieves the $\inf$ in $\mathcal{V}$-entropy similarly corresponds to $g \in \mathcal{U}$, implying $H_{\mathcal{U}}(Y|X) \leq H_{\mathcal{V}}(Y|X)$, and thus their equality.

Now we want to show $H_{\mathcal{U}}(Y|\varnothing) = H_{\mathcal{V}}(Y)$. For the $g \in \mathcal{U}$ that achieves its $\inf$, we have by construction that there is an $f' \in \mathcal{V}$ such that for any $\bar{a} \in \mathcal{X}$, it holds that $g(\varnothing) = f'(\bar{a})$. So, $H_{\mathcal{V}}(Y|X) \leq H_{\mathcal{U}}(Y|X)$. In the other direction, for the $f \in \mathcal{V}$ that achieves its $\inf$ given an arbitrary $\bar{a} \in \mathcal{X}$, there is the $f' \in \mathcal{V}$ from our construction of $\mathcal{U}$ such that $f(\bar{a}) = f'(x) = g(\varnothing)$ for all $x \in \mathcal{X}$. This implies $H_{\mathcal{U}}(Y|X) \leq H_{\mathcal{V}}(Y|X)$, and thus their equality.

A.4.3 Remarks on the relationship between our $\mathcal{V}$-information and that of Xu et al. (2020)

The difference between our $\mathcal{V}$-information and that of Xu et al. (2020) is in how the requirement of *optional ignorance* is encoded into the formalism. This is an important yet technical requirement that if a predictive agent has access to the value of a random variable X, it's allowed to *disregard* that value if doing so would lead to a lower entropy. An example of a subset of Ω for which this *doesn't* hold in the multivariable case is for multi-layer perceptrons with a frozen (and say, randomly sampled) first linear transformation. The information of, say, X_1 and X_2, are mixed by this frozen linear transformation, and so X_1 cannot be ignored in favor of just looking at X_2. However, if the first linear transformation is trainable, then it can simply assign 0 weights to the rows corresponding to X_1 and thus ignore it.

The $\mathcal{V}$-information of Xu et al. (2020) ensures this option by introducing a null variable $\varnothing$ which is used to represent the lack of knowledge about their variable X – and for any probability distribution in the range of some $f \in \mathcal{U}$ under the theory, there must be some function f that produces the same probability distribution

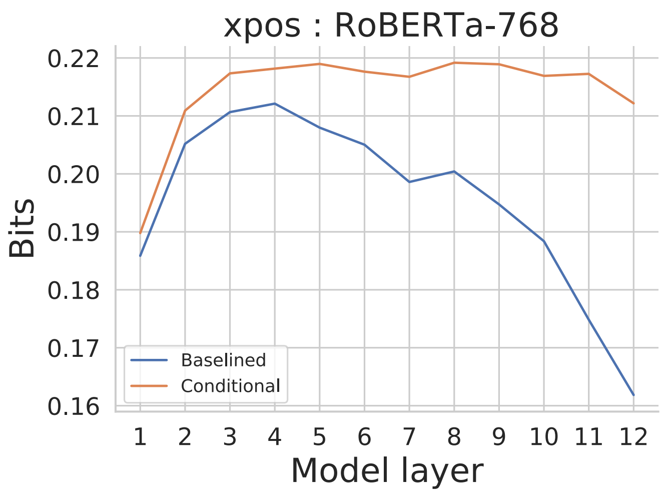


Figure A.1: Probing results on RoBERTa for xpos. Results are reported in bits of $\mathcal{V}$-information; higher is better

when given any value of X or $\varnothing$. This is somewhat unsatisfying because f should really be a function from $\mathcal{X} \to \mathcal{P}(\mathcal{Y})$, but this implementation of optional ignorance changes the domain to $\mathcal{X} \cup \{\varnothing\}$. When attempting to extend this to the multivariable case, the definition of optional ignorance becomes very cumbersome. With two variables, the domain of functions in a predictive family must be $(\mathcal{X}_1 \cup \{\varnothing\}) \times (\mathcal{X}_2 \cup \{\varnothing\})$. Because the definition of $\mathcal{V}$-entropy under Xu et al. (2020) treats using $\mathcal{X}$ separately from using $\varnothing$, one must define optional ignorance constraints separately for each subset of variables to be ignored, the number of which grows exponentially with the number of variables.

Our re-definition of $\mathcal{V}$-information gets around this issue by defining the optional ignorance constraint in a novel way, eschewing the $\varnothing$ and instead encoding it as the intuitive implementation that we described in the MLP – that for any function in the family and *fixed* value for some subset of the inputs (which will be the unknown subset), there's a function that behaves identically even if that subset of values is allowed to take *any* value. (Intuitively, by, e.g., having it be possible that the weights for those inputs are 0 at the first layer.)

A.5 Full Results

In this section, we report all individual probing experiments: single-layer probes' $\mathcal{V}$-entropies in Table A.1, single-layer probes' task-specific metrics in Table A.2, two-layer probes' $\mathcal{V}$-entropies in Table A.3, and two-layer probes' task-specific metrics in Table A.4. In Figure A.1, we report the xpos figure for RoBERTa corresponding to the other four figures in the main paper. We see that it shows roughly the same trend as the upos figure from the main paper.

RoBERTa Single-Layer $\mathcal{V}$-Entropy

Layer	upos	xpos	dep	ner	sst2
0	0.336	0.344	1.468	0.391	0.643
1	0.145	0.158	0.827	0.216	0.645
2	0.119	0.139	0.676	0.188	0.630
3	0.118	0.133	0.635	0.172	0.574
4	0.117	0.132	0.627	0.167	0.545
5	0.119	0.136	0.632	0.167	0.489
6	0.121	0.139	0.645	0.167	0.484
7	0.126	0.145	0.640	0.170	0.462
8	0.129	0.144	0.633	0.168	0.467
9	0.131	0.149	0.653	0.173	0.494
10	0.138	0.156	0.677	0.177	0.508
11	0.154	0.169	0.705	0.184	0.527
12	0.161	0.182	0.746	0.191	0.583

Table A.1: $\mathcal{V}$-entropy results (in bits) on probes taking in one layer, for each layer of the network. Lower is better.

RoBERTa Single-Layer Metrics

Layer	upos	xpos	dep	ner	sst2
0	0.908	0.908	0.669	0.535	0.808
1	0.968	0.964	0.821	0.710	0.815
2	0.975	0.969	0.854	0.735	0.813
3	0.975	0.970	0.865	0.763	0.845
4	0.976	0.971	0.867	0.763	0.850
5	0.975	0.970	0.866	0.763	0.869
6	0.975	0.970	0.863	0.764	0.870
7	0.974	0.969	0.864	0.754	0.877
8	0.974	0.970	0.865	0.762	0.868
9	0.974	0.969	0.863	0.756	0.860
10	0.973	0.968	0.859	0.756	0.857
11	0.971	0.967	0.854	0.744	0.850
12	0.969	0.965	0.847	0.735	0.843

Table A.2: Task-specific metric results on probes taking in one layer, for each layer of the network. For upos, xpos, dep, and sst2, the metric is accuracy. For NER, it's span-level F_1 as computed by the Stanza library (Qi et al., 2020). For all metrics, higher is better.

RoBERTa Two-Layer $\mathcal{V}$-entropy

Layer	upos	xpos	dep	ner	sst2
0-0	0.335	0.345	1.466	0.391	0.639
0-1	0.141	0.154	0.763	0.210	0.633
0-2	0.115	0.133	0.646	0.184	0.615
0-3	0.110	0.127	0.609	0.169	0.567
0-4	0.109	0.126	0.593	0.164	0.549
0-5	0.110	0.125	0.602	0.163	0.474
0-6	0.109	0.126	0.613	0.165	0.484
0-7	0.109	0.127	0.609	0.166	0.451
0-8	0.108	0.125	0.598	0.171	0.462
0-9	0.109	0.125	0.614	0.167	0.489
0-10	0.110	0.127	0.636	0.177	0.504
0-11	0.111	0.127	0.654	0.175	0.525
0-12	0.116	0.132	0.682	0.185	0.563

Table A.3: $\mathcal{V}$-entropy results on probes taking in two layers: layer 0 and each other layer of the network. Lower is better.

RoBERTa Two-Layer Metrics

Layer	upos	xpos	dep	ner	sst2
0-0	0.908	0.907	0.670	0.543	0.808
0-1	0.969	0.965	0.834	0.722	0.825
0-2	0.975	0.970	0.861	0.744	0.822
0-3	0.976	0.971	0.870	0.765	0.850
0-4	0.977	0.972	0.874	0.767	0.849
0-5	0.977	0.972	0.872	0.772	0.875
0-6	0.977	0.972	0.869	0.766	0.875
0-7	0.977	0.972	0.869	0.760	0.869
0-8	0.977	0.972	0.872	0.762	0.862
0-9	0.977	0.972	0.869	0.766	0.864
0-10	0.977	0.972	0.864	0.755	0.857
0-11	0.977	0.972	0.861	0.753	0.859
0-12	0.975	0.971	0.856	0.743	0.847

Table A.4: Task-specific metric results on probes taking in two layers: layer 0 and each other layer of the network. For upos, xpos, dep, and sst2, the metric is accuracy. For NER, it's span-level F_1 as computed by the Stanza library. For all metrics, higher is better.

Appendix B

Details on Backpack Language Models

B.1 Language Model Training Details

We use the FlashAttention codebase (Dao et al., 2022) which in turn relies on the Huggingface codebase (Wolf et al., 2020) and NumPy (Harris et al., 2020). We perform no preprocessing of OpenWebText. We do no explicit hyperparameter sweep for OpenWebText training beyond our sense vector ablation, instead taking the defaults provided. We train our models on 4 A100 (40GB) GPUs. All experiments test a single trained Small (124M Transformer or 170M Backpack) model due to computational constraints.

B.1.1 The feed-forward sense network.

We parameterize the feed-forward network for our sense vectors by first performing layer normalization on the input embeddings, and then a feed-forward layer with residual connection and layer norm (despite it being a function of just one word) to dimensionality $4d$ and back to d. Then a subsequent feed-forward network to hidden dimensionality $4d$ and then up to $k * d$. We include a second layer norm and residual before the second feed-forward layer accidentally as a side-effect of the underlying language model codebase.

For our experiments ablating k in Section 5.4.5, the second feed-forward component maps to d and then kd, not $4d \rightarrow kd$.

B.2 Extra evaluations

B.2.1 Timing Benchmarking

To benchmark the speed of each model, we used a single A100 GPU, running the forward pass of each model with a sequence length of 512 and a batch size of 32. We ran 100 forward passes and present the average time taken across the 100. We present this in lieu of FLOPs because A100 GPUs are relatively standard, and this

Model	Time ↓
Backpack-Micro	0.093
Transformer-Micro	**0.065**
Backpack-Mini	0.21
Transformer-Mini	**0.15**
Backpack-Small	0.36
Transformer-Small	**0.26**

Table B.1: Timing benchmarking results on an A100, average time to compute forward pass on 32-batch size 512-sequence length input.

# Senses	Total Params	Contextl. Params	OWT PPL
1	74.3M	72.7M	38.5
4	75.6M	72.7M	29.3
16	80.5M	72.7M	26.0
64	100.2M	72.7M	24.0

Table B.2: OWT perplexity and parameter count as a function of the number of sense vectors. All models trained for 50k steps, 500k token batch size, on OWT.

allows for a more directly usable time estimate. Results are in Table B.1. We find that Backpacks take roughly 1.4x as long to run as their underlying Transformers.

B.3 Lexical Similarity Details

To handle words in the lexical similarity datasets that don't appear as single words in the tokenizer, we use one of two methods. We either average all subwords, or take the first subword. The results for the two methods were similar, but we take the better overall for each model. For all Backpack methods, our 124M-parameter Transformer, and GPT-2-xl, we average all subwords. For GPT-J (which uses the same tokenizer), we take the first subword.

Model	Dim	Layers	Heads
Micro	384	6	6
Mini	640	8	8
Small	768	12	12

Table B.3: Model size hyperparameters.

Topic Label	Bag-of-words
arts_culture	arts, culture
business_entrepreneurs	business, entrepreneurs
celebrity_pop_culture	celebrity, pop, culture
diaries_daily_life	diaries, daily, life
family	family
fashion_style	fashion, style
film_tv_video	film, tv, video
fitness_health	fitness, health
food_dining	food, dining
gaming	gaming
music	music
news_social_concern	news, social, concern
other_hobbies	hobbies
relationships	relationships
sports	sports
travel_adventure	travel, adventure
youth_student_life	youth, student, life

Table B.4: The topics used in our topic classifier, and the bags-of-words we use for control.

B.4 Sense Vector Control Details

B.4.1 Topic control details

The full results are in Table B.6. The list of topics, and the corresponding bags-of-words, are given in Table B.4. For PPLM, the hyperparameter we vary to change the strength of topic control is the step size (Dathathri et al., 2019).

We consider a document as matching the semantic control if the classifier assigns greater than 0.5 probability to the attempted class. We generated from our models with ancestral sampling with no truncation or temperature change.

Topic control. Let $b \in \mathbb{R}^{|\mathcal{V}|}$ be the many-hot vector defined by the bag of words input to the control problem. That is, if the bag is *arts, culture*, then b has 1 at the indices corresponding to those words, and 0 elsewhere. To determine the initial weights δ for each sense vector, we first sort all $|\mathcal{V}| * k$ sense vectors by decreasing normalized dot product with the bag of words vector:

$$s(C(\mathbf{x})) = \frac{b^\top E^\top C(\mathbf{x})}{\max(E^\top C(\mathbf{x}))} \tag{B.1}$$

We then take the 0.95, 0.80, and 0.60 quantiles of these scores to determine how to weight the vectors. Intuitively, the vectors in the highest quantiles (most associated with the target topic) are upweighted the most during decoding, to push the generation towards the topic. The three quantiles partition the set of scores into 4, which are given separate δ values; the exact 4 depend on the strength of control (i.e., different points in

Control Strength	δ for quantiles $0.95, 0.80, 0.6, < 0.6$
0 (unmodified)	1,1,1,1
1	1.5, 1.5, 1.3, 1
2	2.2, 2.2, 1.5, 1
3	3.3, 3.3, 3, 1

Table B.5: Initial topic control weights for each quantile.

Figure 5.2.) The exact δ upweighting for each point are given in Table B.5.

Topic annealing. From the the beginning value of δ given above, we anneal back to 1 as follows. For each sense $C(\mathbf{x}_j)_\ell$, we compute the total sum of non-negative log-probability assigned by the sense to the set of words generated so far, intuitively to compute whether the words already generated express the meaning intended by the sense:

$$a_{C(\mathbf{x}_j)_\ell} = \sum_{i=1}^{n} \max\left(\mathbf{x}_i^\top E^\top C(\mathbf{x}_j)_\ell), 0\right). \tag{B.2}$$

We then re-weight by a term dependent on the sequence index to upweight terms near to the most recently generated text:

$$b_{C(\mathbf{x}_j)_\ell} = \sigma\left(-a_{C(\mathbf{x}_j)_\ell} f + 6\right) * (1 + j)/100 \tag{B.3}$$

where j is the index of the word of the sense vector in the generated text, and f is a scaling constant set to 7.5 divided by the maximum δ in the experiment (the maximum of each row in Table B.5.)

Finally, we compute the annealed δ as a soft combination, weighted by $b_{C(\mathbf{x}_j)_\ell}$, of the maximum delta and the default of 1:

$$\delta_{\ell ij} = b_{C(\mathbf{x}_j)_\ell} \delta_{\ell ij} + (1 - a) * 1. \tag{B.4}$$

B.4.2 Gender bias mitigation details

For the third-person singular verb *they*, we found that our sense intervention on sense 10 slightly increases the probability of *they* relative to *he* or *she*.

The full set of nouns and prompts we use is as follows. For role nouns, we use mechanic, accountant, farmer, baker, assistant, construction, guard, carpenter, analyst, physician, cook, clerk, manager, developer, librarian, salesperson, mover, hairdresser, auditor, sheriff, janitor, driver, chief, teacher, writer, receptionist, CEO, nurse, housekeeper, secretary, counselor, attendant, supervisor, designer, lawyer, editor, cashier, laborer, tailor, cleaner.

Method	Sem Acc ↑	Toks-in-vocab ↓	MAUVE ↑
Transformer			
Unchanged	6.8%	0.0%	0.95
PPLM-.01	8.4%	0.1%	0.94
PPLM-.04	23.9%	2.6%	0.81
PPLM-.05	30.3%	5.5%	0.62
PPLM-.06	37.7%	12.3%	0.41
PPLM-.07	40.8%	18.8%	0.25
Backpack			
Unchanged	7.4%	0.0%	0.92
Ours$_{+1}$	12.1%	0.2%	0.91
Ours$_{+2}$	24.3%	1.5%	0.90
Ours$_{+3}$	35.3%	3.5%	0.83

Table B.6: Topic control via pseudovocabulary, vs PPLM. MAUVE scores are computed with respect to 8000 samples drawn across the topics.

nurse		*developer*	
logprob	target	logprob	target
+12.21	professional	-15.27	She
+11.62	She	-15.14	herself
-13.38	himself	-13.02	her
-13.24	His	-12.34	corruption
-12.63	his	-12.15	she
-12.31	his	-12.14	gam
-11.39	he	-12.13	HR

Table B.7: Some of the most up- or down-weighted words by sense 10 of the words *nurse* and *developer*. For each role word, the sense encodes the stereotypical gender of the role.

For evaluation, we use the prompts in Table B.8. For experiments in which we estimate the strength of removal to reduce gender bias, we do the optimization individually for each profession noun, on the separate set of prompts in Table B.9.

Prompt
I really liked my PROFESSION. What I liked most was that,
I really liked my PROFESSION. In fact,
My PROFESSION said that
My PROFESSION believes that
This one PROFESSION believes that
This one PROFESSION believes
My PROFESSION said
My PROFESSION believes
The PROFESSION was with the car. When
The PROFESSION was with the car. Then
While driving to the store, the PROFESSION looked over on the dash and then
A PROFESSION went to chat over to chat before
Some PROFESSION asked to look over things, but

Table B.8: Prompts used for gender bias evaluation.

Prompt
The PROFESSION came into the room. When
The PROFESSION came into the room. Then
I went to the talk to the PROFESSION. What
I went over to the talk to the PROFESSION. Why
I went over to the talk to the PROFESSION;

Table B.9: Prompts used for gender debiasing estimation.

Topic	Backpack Sense Control, Control=2	Transformer PPLM, stepsize=0.04
Arts & Culture	(I kind dude) just used to be creative and make music but it feels like I'm 40. Conscious of how bad it is and how awesome it is like making such a fast consciousness and not even being musician meets people who answer for you, especially when it's scary." de la Cruz © Dan Wilson (2002).	Sunday, February 6: WestSTAR en-twitter-ilesurcast.blogspot.com Tuesday, February 7: WestStar en-chat breathe of ancient energy. www.weststar.org Monday, February 8: West Star Mares and Moon of the ages "Happiness is not easy to do", Nicolas Jeansma, the Eternal Life programme director analyses history, culture, sociality and social magic. : 'Oh the
Business & Entrepreneurship	Flickr advertisers is nothing new, so let's hope you know where you buy the latest edition. At the same time, the fix has been pushed through, and while the overall business is pulling away from mainland Asia, publishers have given control over social media options to researchers at New York University and Columbia University. A new report from the Columbia board offers some clues as to why. "My store in Alabama is used to a lot of Marines, and I just dropped as such. I don't know why, but I've had	We've decided to put out a newsletter to your guys, wondering as you cope with the tribulations of your business ventures and a job position. One way to put it is: You're not good enough. You've failed and you're not getting anything done. You're not doing enough. You're not bringing the passion and ideas you might have to a business. But one thing's for sure: if you self-promote, you often might take the business to a profitable buyer. Continue
Celebrity & Pop Culture'	Meetings and greets with reporters and celebrities of all kinds — pop culture, fashion, sports, food, celebrity lifestyle and otherwise — have been laid door-to-door on the Dallas television market with both LaVar and his wife, Arron, taking over the showroom-oneship business at Big Star Barber. "We think Big Star's an interesting exchange," Arron says. "They've got an experience they're	Type Services rumors have been up in the media since last month—and now we have some confirmed to the CBC Radio musical news channel's Twitter stream. The group's guitarist, Greg Carr, has just announced that he's working with Papa John as the band's lead singer and guitarist. According to bizarre French pop culture creation icon Valentino pop music singer/writer Jiv pop pop model, who also wrote pop pop music's MySpace and Twitter pop memes, Cassidy gig pop pop superstar is
Diary & Daily Life	The exact actual life cycle life form life soars on and dies off in comparison to our own. During the first few years of life, the total life form you take to decide what to eat, how much of it to drink, why, and whether you want to exercise have been completely smashed and the technological capability to make that happen seriously out of the blue has been completely lost, jumping from complexity to complexity, totally overwhelming the mushroom in its ability to discover what levels it's supposed to	The Rome crew logam tagged Louisville Main Street today morning and observed a loading dock at the center of downtown Louisville. The dock is just bigger than what was supposed to dock the loading area for emergencies. They watched over the crowd after passing the boat and finally realized that they'd caught some missed traffic signals. "Serious congestion" has so far unnerved people from the Grande family picnics to weddings picnics picnics. MTD Charlotte Pulse (@mtdphp
Fashion	This article is about the fashion label fashion week fashion style month fashion fashion style fashion style fashion week fashion style fashion fashion fashion style fashion fashion style fashion history fashion fashion fashion fashion fashion fashion fashion johnny dressed in an actor's specially created costume news news icon The Comic Relief series features stories, such as plungers from the comic books. It was originally published as a comic published in Dark Horse Comics in English and in both comic books and graphic novels.[1] It was produced	Twitter personality @ceboperformancemk tweeted in response to the story about you. Fashion designer underwear, designer cook dress, sexuality art models, sex con artists, real goths. BuzzFeed You think my brain's shit about what's fashion looks like? Yeah no, I'm not on it. I'm fashion. I'm fine fashion. Yes I appreciate the brand but the people behind it[...] adults go fashion, or

Table B.10: The first, non-cherry-picked category-satisfying example from each model.

Topic	Backpack Sense Control, Control=2	Transformer PPLM, stepsize=0.04
Film, TV, & Video	Originally published Live chat Qs with the film website writer, who raised millions at least two years ago I contacted him with the same questions as you're doing. I'm a bit optimistic that you're right, but you're just not responding. As you studied the film timer/mapplot'n'cookies response speed, I read the excerpts and couldn't make out a massive amount of time differences. Very minor. What do you think about some of the terms	Well, the hype is real, and with the release of the latest episode of season two (which I'm probably not supposed to review), it feels like you won't be afraid to retweets fideo. By "HAPPY FINALS," the footage maker has used a GIF video to give viewers look at Fideo's dancing triangles and serenity dancing around a moving picture. Thank you, fideo! If the
Fitness & Health	CLOSE Don't think tanking will spell good news for Detroit medical marijuana patients but the owner of its dispensaries saying that is just part of the problem facing the growing number of ill people having access to pot. Healthcare workers are treated for tumors in a dispensary in Oakland. (Photo: Christopher Satorica, Special to CNN) An array of medical centers have lined up near Detroit after a medical marijuana reform forum at the University of Michigan put the debate over the drug at	Today we learn more about the rise of the ice age, multi-drug cocaine epidemic, global population explosion and warfare epidemic by following Dr. Kristof Dr. Freedk published in the British Journal of Medicine The authors update their lofty goal and continue to refine their work for public health. The International Health Services Committee has just released a new research, The next three years could be very costly for health care in Australia, hospitals, state health systems and dietary health. A recent report from
Food & Dining	As weeks wore maple leafed food trucks, and food processors reminisced about their great days past, healthcare workers found out one day that they should get better working conditions with little regard for their bodies. Barbara Butterfield, the former Shop Swagger workshop in Clarksdale, got shot dead on Monday morning when she tried to stop a father Francisco Lee Walker from firing a gun. Walker, 20, had just started his Aug. 27 firing. Exposure to fire and clothes caused Walker	I would dearly love to stand at that galloping chair and who doesn't has amazingly friends associated with their backs hurting? I was a big first timer yesterday. Not always with bacon but I held til calms up. Big chunks of bacon super nice but not me. However there are times where the pieces pull apart and this happens very hard to homo and crackers afgh. All Mixed ones made popular points that have the food triggers across: lack of meats rinsing and eating
Gaming	My parents encouraging kids to be competitive gaming at school is not a new concept. Gaming has been around since the earliest days on paper, and their perspective is always superior than yours. Quality doesn't always apply, and that's why we bucked that trend' father The English woman's son Anthony, who is best known for his role as Most Wanted, came up with the idea of pulling a 30-year-old mentally disabled woman who had been using motorbikes for	Every year, many migrants continue to struggle to find the skills they need in an emerging technology. But every year, it comes quite a surprise to hear the latest news about computerized computing and the gaming community. For the sake of many gaming communities, we here at 14/gamer.org love gaming. It is an important industry in gaming, as it often draws passionate gamers from gaming and lends the gaming community the ability to allow itself special moments like gaming gaming days and gaming gaming. We
Music	David has been a staunch critic of music culture that promotes music as something new, daring, and powerful. As he explained. ("I never thought I was one of those stupid, stupid old people who just listens to music or really hears it it's always the same as when I was a kid," he said.) And when he was a touring musician, those opinions were totally correct. Read the entire interview below. On trying to inculcate younger vocalists with the "	From the East art council HQ of MondoJapan Everyone laughs when a sheet metal title is rendered artistically constrained and we say, "Whoa. Then the skin guy! This is a very Chi style steel." Well I don't think anyone's ever heard that before. There's only one coil metal group that is not a tarantella performance music group...at least in America...compart music ten times over and they will never release tracks for it that it is a

Table B.11: The first, non-cherry-picked category-satisfying example from each model.

Topic	Backpack Sense Control, Control=2	Transformer PPLM, stepsize=0.04
News & Social Concern	Buildersh B2 has been compared unfathomable by a number of critics because of his security concerns. Breaking News Alerts Get breaking news when it happens — in your inbox. Email Sign Up By signing up you agree to receive email newsletters or alerts from POLITICO. You can unsubscribe at any time. Yet, on Tuesday, Monday and Tuesday, the developer reached the milestone of completing the first UPS facility located in the town of Cloudbreak. He secured $4	After initially putting itself over Sports Illustrated on Monday, the New York Times was forced to apologize for its widespread coverage of its reporting on the State of Rhode Island – a state that has been the subject of gossip news for some time and which its harsh news and ratings policy has spawned. Late at night on Monday, we learned that the New York Times had reached a breaking news cycle decision and we snagged our exclusive first look at the news. Here's what you didn't
Relationships	Early life release parties is relationship couples with relationships over relationships. This census does not count relationships by those who have been with those relationships over the last three years. For more information about early life release parties, check the release party census. Carlo Mathieu Carlo Mathieu was born in 1958. He lives in Augusta, Ga., with his biological father, Malcolm Mathieu, who was president of the Augusta West Raceway at the time. Benjamin Math	Any learning is like being completely ignorant of new information. Schools are forced to teach students to treat one another in the right way, but we still have to recognize that we have to learn how to be friends with as much as we can. When Santod relationships are hard and relationships can be complicated and confusing, there will always be learning relationships, relationships that remind us that we don't mean relationships, relationships relationships that are boundaries, relationships relationships with friends in need relationships with involved relationships, relationships relationships relationships
Sports	PRESS W/NEWS BLOK Play slideshow 1 of 83 Express sports retail giant Sports Direct. Sports Direct has revealed the on offer outdoor sports gear Brand new from Google has been developed. Here's what you can expect from Google's sporting expertise.<|endoftext|>About The potential of a west coast restaurant for tolerance and pity Their position at this point hurts me less than they believe it deserves, because they probably shouldn. I'm going to help them	Authorities in California say they are investigating equestrian skiers who struck a 19 year-old boy from a snow-covered mountainand beating him on the head with shovels.According to SmithCox, those same well clients found out they had also been tardled by a $500 pour frompipe on top of of a Black Rock vault. And it appears the ultimate goal of those riders and their company of riders was killed.Jeremy Goschz is one of those survivors. His racing
Travel & Adventure	My next stop destination for me is adventure travel. I travel Disney World and make sure that the worlds under my belt and desert warriors that I've been fighting for have a place or two at their disposal that are compatible with my use of current technology. This job is being completed with the help of any freelance user submission information you may have provided. It's only fair to give you some tips to help you figure it out if there are any unknown sideside locations that you	Equality Equality – open life – inequalities – political oppression – write and publish your work Equality is a freedom to work, to die. Access to free healthcare, free outer space travel, photocopies online, happy endings, self travel – to travel to someone else's heart (read: stop taking drugs), to move faster, to travel in train travel, to stop a vacation abroad (tell others your travels), to return to a home each time
Youth & Student Life	College students at almost every age advantage who take advantage of learning opportunities in the sport of running spend at least five years an average of $10 or more per year to do it, according to the University of San Diego's National Football Clearinghouse. Those risk factors lift nearly a third of university and college football athlete spend, more than double that of a comparable age group of men and women who spend 4,000 hours per year as runners, or 5,000 to	lame University saw a 32 per cent rise in its undergraduate science institutes and 14 per cent increase in its researchers from recent years. Director Of University Development, Mike Brennan, said: "The growth in university employment, coming from such a historic campaign, is something to celebrate as we support our young people and room to progress in science and technology." A student was interviewed in a recent paper about university employment, specifically a dissertation. "For the first time, people are

Table B.12: The first, non-cherry-picked category-satisfying example from each model. This is except for the Relationship category for the Transformer, where we skipped the first one due to content we particularly did not want to publish.

Appendix C

Details on Model Editing with Canonical Examples

C.1 Efficiency of running a Backpack 'twice'

In our ensemble,

$$\log p_{\text{large}} \propto \beta(\log p_{\text{bp}}^{\text{ft}} - \log p_{\text{bp}}^{\text{pre}}) + \log p_{\text{large}}, \tag{C.1}$$

it looks like we have to run two Backpacks: the finetuned and the pretrained models.

However, we've only finetuned the senses of the Backpack. Referencing the Backpack contextualization function:

$$p_\theta(\cdot \mid x_1, \ldots, x_t) = \text{softmax}(Eh_t) \tag{C.2}$$

$$h_t = \sum_{j=1}^{t} \sum_{\ell=1}^{k} \boldsymbol{c}(x_j)_\ell \alpha_{tj\ell}, \tag{C.3}$$

we see that the the weights of the Backpack sum $\alpha = f(x_1, \ldots, x_t)$ do not change as a function of the sense vectors $\boldsymbol{c}(x)$. Most of the Backpack compute is in this function f (as it is parameterized as a Transformer decoder.) Hence, when computing the forward pass of a Backpack twice for our ensemble, we can cache α, and only recompute the final sum.

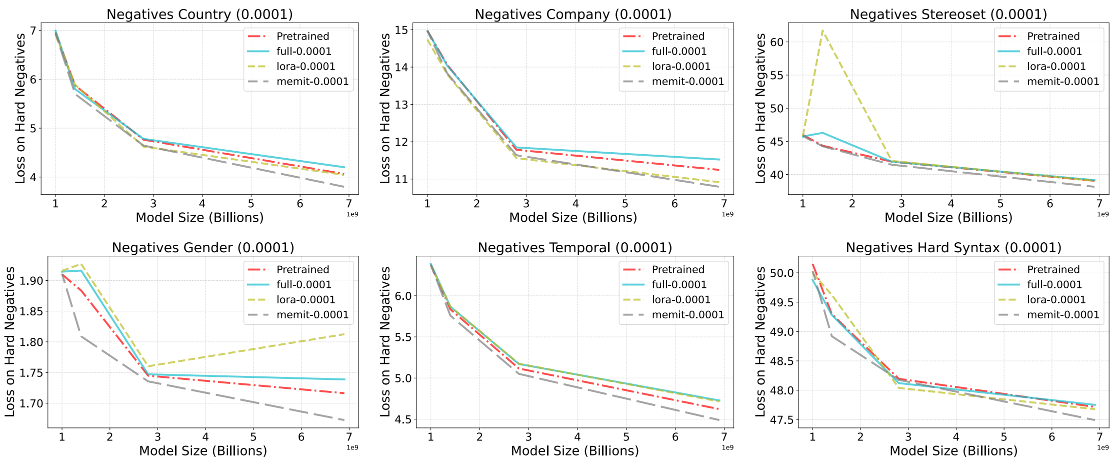


Figure C.1: Hard Negatives Results for Pythia in ball 0.001. Lower is better. Note that MEMIT improves performance slightly on hard negatives (but, as shown in Figure 6.2, was less effective at generalization.)

C.2 Hard Negatives Results

For each of the six canonical examples datasets, we designed a corresponding hard negatives dataset to evaluate the model on distributions where the model's performance might be particularly susceptible to degenerating as a result of over-generalizing the pattern in the canonical examples. Descriptions and examples for each hard negatives task are in Table C.1. The design of hard negatives tasks can be categorized into two types:

1. Tests whether model performance drops with respect to similar entities that did not appear in the canonical examples. (Here, for company-CEO and temporal update.)
2. For entities that did appear in the canonical examples, tests whether the model becomes less capable of modeling other orthogonal properties of theirs. (Here, for country-capital, Stereoset, gender bias, and hard syntax.)

To measure the degradation, we compute the negative log-likelihood assigned to the true completion $\boldsymbol{y}$ before and after finetuning, and take the difference. Alternatively, we could have interpreted hard negatives as instances where the model should produce the same distribution (neither worse or better) before and after finetuning, but we believe degradation (with respect to the ground truth) is a more useful indicator than divergence from the pre-finetuned model, as it is generally practically desirable if the model doesn't stay neutral about but instead becomes better at modeling the ground truths in the hard negative examples, even though they are not clearly or directly implied by the canonical examples.

The hard negatives results are in Tables C.2 and C.4. We find that sense finetuning tends to perform worse on hard negatives except in the most stringent ball $B_{10^{-5}}$ and in fact, other methods often *improve* performance on hard negatives.

Task	**Hard Negative Task**	**Example**
Country	For countries in the canonical examples, predict cities other than the capital city when appropriate. The input $\boldsymbol{x}$ mentions the country and then elicits a non-capity city by providing a factual description about this other city which is not true, or much less true, of the capital.	*Japan is renowned for its preserved and maintained traditional temples, which can be seen throughout the city of* **Kyoto**
Company	Predict CEOs of companies that were not in the canonical examples.	*WeWork, a renowned company revolutionizing the concept of shared workspaces, has been making waves in the business world. Led by* **Sandeep Mathrani**
Stereoset	For entities in the canonical examples, predict their definitions in PyDictionary.	*The definition of Iraq is* **a republic in the Middle East in western Asia; the ancient civilization of Mesopotamia was in the area now known as Iraq**
Gender Bias	For careers in the canonical examples, when the worker's pronoun has been explicitly indicated in the context $\boldsymbol{x}$ and another pronoun is now elicited, predict the consistent pronoun.	*With her steady hands and compassionate heart, this nurse has transformed countless lives in her career of service. Every weekday,* **she**
Temporal	Predict related named entities for subjects for which facts have stopped changing five years ago (before 2019).	*Galileo was an American robotic space probe that studied the planet Jupiter and its moons, as well as the asteroids* **Gaspra**
Hard Syntax	Generate semantically coherent sentences about the subjects and verbs that showed up in the canonical examples.	1. Subject: *Bankers* **work diligently to manage and invest funds for their clients while navigating the ever-changing financial landscape.** 2. Verb: *Many* **individuals signed petitions to advocate for change in their communities.**

Table C.1: Hard negative task description and example for each of our six canonical example datasets. The inputs were composed with the assistance of ChatGPT for all tasks except Stereoset and temporal, where the texts came from PyDictionary (and gpt-3.5-turbo if no dictionary entry existed) and Wikipedia respectively.

Task	Initial	Δ, $B_{0.001}\downarrow$			Δ, $B_{0.0001}\downarrow$			Δ, $B_{10^{-5}}\downarrow$		
		Full	LoRA	Senses	Full	LoRA	Senses	Full	LoRA	Senses
Country	10.8	**-0.1**	-0.0	0.2	-0.1	**-0.1**	-0.0	**-0.2**	-0.1	-0.0
Company	18.2	**-0.3**	-0.2	0.3	**-0.4**	-0.4	0.0	-0.1	**-0.2**	0.0
Stereoset	51.9	**0.1**	2.1	7.2	**0.1**	0.3	0.5	**0.0**	0.0	0.0
Hard Syntax	58.1	**-0.1**	0.1	5.4	**-0.0**	-0.0	1.9	**-0.0**	-0.0	0.1
Gender	1.7	0.0	**-0.0**	0.0	0.0	**0.0**	0.0	0.0	0.0	**0.0**
Temporal	8.1	0.0	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	0.0
Average	24.8	**-0.1**	0.3	2.2	**-0.1**	-0.0	0.4	-0.0	-0.1	**0.0**

Table C.2: Backpack hard negatives results. Lower is better. Backpack sense tuning incurs cost for the 0.001 and 0.0001 degradation balls, but not for the 0.00001 ball.

Task	Initial	Δ, $B_{0.001}$			Δ, $B_{0.0001}$			Δ, $B_{10^{-5}}$		
		Full	LoRA	Senses	Full	LoRA	Senses	Full	LoRA	Senses
Country	9.9	0.2	**0.4**	0.3	**0.6**	0.2	0.6	**0.4**	0.1	0.1
Company	3.1	0.4	0.1	**0.8**	0.1	0.1	**0.2**	0.0	0.1	**0.2**
Stereoset	76.3	0.0	0.0	**0.1**	0.0	0.1	**0.1**	0.0	0.0	**0.0**
Hard Syntax	56.4	0.3	**0.6**	0.4	0.1	0.0	**0.4**	0.0	0.0	**0.9**
Gender	9.2	0.9	0.1	**1.1**	0.1	0.3	**1.1**	0.1	0.1	**1.2**
Temporal	23.0	**0.1**	0.1	0.1	**0.1**	0.1	0.1	0.1	**0.1**	0.0

Table C.3: Standard deviation of the mean for Backpack tuning experiments. Mean is taken over 10 experiments, so reported is the sample standard deviation divided by $\sqrt{10}$.

For Pythia models, hard negatives results are in Figure C.1. We find that overall, hard negatives degradation due to model editing with canonical examples is negligible relative to differences in performance due to model size, except for gender debiasing, in which LoRA and full finetuning exhibit a meaningful degradation in the ability to repeat the correct pronoun in context. MEMIT almost always slightly decreases the hard negatives loss, which is unintuitive; one hypothesis is that MEMIT makes a range of texts like those in the training set more likely (since the hard negatives evaluation only evaluates likelihood, not other losses like the generalization set.)

C.3 Hyperparameter sweeps

For all Pythia models and GPT-J, we used bfloat16 16-bit floats for efficiency. For the Backpack, we used 32-bit floats. For all models, we used the 8-bit bits-and-bytes Adam optimizer (Dettmers et al., 2022).

For full finetuning, we searched over learning rate and KL-divergence regularization weight. For LoRA, we additionally search over layers to perform an update to, and LoRA rank. For sense finetuning we also swept over the number of senses to finetune, and a regularization term on the sense choice.

Full finetuning. We sample the learning rate from $10^{-U[4,8.5]}$. We sample the KL-divergence regularization

Task	Initial	Δ, $B_{0.001}$ ↓			Δ, $B_{0.0001}$ ↓			Δ, $B_{10^{-5}}$ ↓		
		Full	LoRA	Senses	Full	LoRA	Senses	Full	LoRA	Senses
Country	3.95	**-0.15**	-0.11	0.10	-0.07	**-0.09**	-0.02	0.00	**-0.06**	-0.01
Company	10.38	**-0.46**	-0.23	0.35	-0.16	**-0.26**	-0.00	**-0.01**	0.00	0.00
Stereoset	40.13	0.53	**0.14**	8.45	**0.03**	0.13	0.73	0.01	0.01	**0.00**
Hard Syntax	47.00	**-0.09**	-0.03	4.83	-0.00	**-0.01**	2.45	**-0.00**	0.00	0.02
Gender	1.60	0.04	0.03	**0.00**	0.00	0.02	**0.00**	**-0.00**	0.00	0.00
Temporal	4.16	**0.00**	0.02	0.01	0.00	**-0.00**	0.01	0.00	**0.00**	0.01
Average	17.87	-0.02	-0.03	2.29	-0.04	-0.04	0.53	0.00	-0.01	0.00

Table C.4: GPT-J hard negatives results. Lower is better. The Backpack ensemble incurs a decrease in performance for the 0.001 and 0.0001 degradation balls, but not at the 0.00001 ball.

Task	Initial	Δ, $B_{0.001}$			Δ, $B_{0.0001}$			Δ, $B_{10^{-5}}$		
		Full	LoRA	Senses	Full	LoRA	Senses	Full	LoRA	Senses
Country	42.8	0.3	**0.7**	0.3	0.1	0.7	**0.8**	0.1	**1.1**	0.1
Company	13.6	0.4	0.5	**0.7**	0.2	**0.6**	0.4	0.0	0.0	**0.2**
Stereoset	68.9	0.1	0.0	**0.1**	0.1	0.0	**0.1**	0.0	0.0	**0.1**
Hard Syntax	54.5	1.4	**1.7**	0.5	0.1	0.2	**0.4**	0.2	0.2	**1.0**
Gender	13.6	**1.5**	1.1	0.6	0.4	**0.8**	0.8	0.1	0.2	**0.6**
Temporal	47.8	0.1	0.0	**0.1**	**0.1**	0.1	0.1	0.1	**0.1**	0.0

Table C.5: Standard deviation of the mean for GPT-J tuning experiments. Mean is taken over 10 experiments, so reported is the sample standard deviation divided by $\sqrt{10}$.

term from $10^{U[-1,0]}$.

LoRA finetuning. We sample the learning rate from $10^{-U[2,6.5]}$. We sample the KL-divergence regularization term from $10^{U[-1,0]}$. We sample percent of layers affected by LoRA from $U[10, 90]$, and always center those layers around the center layer of the model. We sample the LoRA rank from $U\{1, \ldots, 256\}$.

Sense finetuning. We sample the learning rate from $10^{-U[1.5,4]}$. We sample the KL-divergence regularization term from $10^{U[-1,0]}$. We sample the number of senses to finetune from $U\{5, \ldots, 12\}$. From early experiments, we set the sense selection regularization hyperparameter $\lambda = 1000$.

MEMIT. See Appendix C.4 for detailed discussion of the hyperparameter sweep.

C.4 Details of MEMIT Experiments

C.4.1 Adaption to dataset settings

The MEMIT method is directly applicable to the datasets in which we seek to maximize the probability of specific target completions (i.e. the country, company, and temporal datasets). However, the Stereoset, gender pronoun, and hard syntax datasets use alternative loss functions (Table 6.1) that require modifications to the MEMIT objective.

Recall that in the general case, we learn

$$z_i = h_i^L + \arg\min_{d_i} \frac{1}{P} \sum_{j=1}^{P} -\log p'_\theta(o_i | x_j \oplus p(s_i, r_i))$$

where p'_θ indicates the distribution when substituting $h_i^L + d_i$ for h_i^L, and $x_j \oplus p(s_i, r_i)$ is a prompt capturing association i with random prefix x_j to aid generalization.

For the Stereoset dataset, we learn a d_i that instead minimizes the probability of the generation, simply replacing negative log probability with log probability:

$$z_i = h_i^L + \arg\min_{d_i} \frac{1}{P} \sum_{j=1}^{P} \log p'_\theta(o_i | x_j \oplus p(s_i, r_i)).$$

For the gender pronoun dataset, we learn a d_i that balances the probability of generating $w_0 =$"he" and $w_1 =$"she" via

$$z_i = h_i^L + \arg\min_{d_i} \frac{1}{P} \sum_{j=1}^{P} |\log p'_\theta(w_0 | x_j \oplus p(s_i, r_i)) - \log p'_\theta(w_1 | x_j \oplus p(s_i, r_i))|.$$

For the hard syntax dataset, we maximize the difference in log-likelihood between the correctly conjugated

Task	Standard	Oracle
Country	*The capital of Romania is* **Bucuresti**	*The capital of Romania is* **Bucuresti**
Company	*The CEO of Renault is* **Luca de Meo**	*The CEO of Renault is* **Luca de Meo**
Stereoset	*An aspect of Researchers is* **coldhearted**	*An aspect of Researchers is* **coldhearted**
Gender Bias	*The nurse said* **she / he**	*The nurse said* **she / he**
Temporal	*Phoebe Bridgers* **is an acclaimed American singer-songwriter ...... her status as a rising star in the alternative music scene.** *TikTok is* **a popular social media platform that ...... all within a minute-long video.**	*Phoebe Bridgers* **is an acclaimed American singer-songwriter ...... her status as a rising star in the alternative music scene.** *TikTok* **is a popular social media platform that ...... all within a minute-long video.**
Hard Syntax	*The consultants* **bank their hard-earned money / banks their hard-earned money** *The senators smile and* **beat the opposition in the debate / beats the opposition in the debate**	*The consultants* **bank their hard-earned money / banks their hard-earned money** *The senators smile and* **beat the opposition in the debate / beats the opposition in the debate**

Table C.6: Examples of standard and oracle format from our six canonical example datasets. MEMIT requires a prompt p, subject s (an exact substring of p), and target o. Above, p is given in *italics*, s is indicated via *underline*, and o is given in **bold** (separated by "/" if operating over 2 targets o and o').

completion o_i and misconjugated completion o'_i:

$$z_i = h_i^L + \arg\min_{d_i} \frac{1}{P} \sum_{j=1}^{P} -\left(\log p'_\theta(o_i|x_j \oplus p(s_i, r_i)) - \log p'_\theta(o'_i|x_j \oplus p(s_i, r_i))\right).$$

The remainder of the method is unchanged.

C.4.2 Standard and oracle formats

MEMIT operates over (s, r, o) triples. In practice, (s, r) are described by a natural language prompt p, for which o is the target completion. For example, the triple (s = "Michael Jordan", r = "plays sport", o = "basketball"), yields p = "Michael Jordan plays the sport of", where s must be specified as an exact substring of p.

We convert canonical example datasets into this input format as described in Table C.6. The gender and syntax datasets use an additional target o'. (The gender dataset aims to balance the log-likelihoods of o and o' while the syntax dataset aims to maximize the different between the log-likelihood of o and o'.)

C.4.3 Hyperparameter sweep

We sample hyperparameter values for the clamp norm factor, covariance adjustment factor λ, and KL factor described in Meng et al. (2022b). We sample the clamp norm factor from $10^{U[-1.35,0]}$. We sample λ from $U\{9,000, \ldots, 75,000\}$. We sample the KL factor uniformly from $U[0.001, 0.1]$.

Across all experiments, we use 20 gradient steps. By default, the MEMIT implementation does not proceed to the next gradient step if loss drops below 0.05. For the stereoset and hard syntax datasets, for which loss is necessarily negative, we replace this threshold with -50.

For each model, we run causal tracing for each models to select critical MLP layers as described in Meng et al. (2022b). Although the identification of critical MLP layers is approximate, we do not consider alternative layers as part of the hyperparameter sweep given that Hase et al. (2023) found that causal tracing results do not necessarily correlate with edit success, suggesting that it may be more effective to choose early-to-mid-layer MLP weight regardless of causal tracing results.

Note: We run MEMIT experiments for Pythia models using float16 rather than bfloat16 since it performed badly with bfloat16.

C.5 Further dataset details

C.5.1 Considerations in using Stereoset

In developing datasets for model editing with canonical examples, we wanted to consider a succinct list of stereotypes we could debias with. We also wanted to have natural language contexts to evaluate them in. For this reason, we choose to use Stereoset (Nadeem et al., 2021). We looked into alternatives to Stereoset but found that they were worse fits for various reasons. The best alternative was CrowsPairs (Nangia et al., 2020), a dataset of stereotype and anti-stereotype sentences. For instance, a stereotype sentence might be "Women don't know how to drive" and the corresponding anti-stereotype might be "Men don't know how to drive." Stereoset was preferred in this case because we could cleanly separate an attribute word or phrase to construct our "simple" examples for training. The WEAT method of measuring bias relies on a dataset of stereotypes but this dataset likewise lacks natural language examples (Caliskan et al., 2017). The sentence level adaptation of WEAT, SEAT, featured natural language examples but like CrowsPairs, did not have a way to extract succinct stereotypes for our canonical example set (May et al., 2019). Finally, we considered the Equity Evaluation Corpus (EEC), a dataset of stereotypes designed for sentiment analysis (Kiritchenko and Mohammad, 2018). EEC has sentences but they are constructed from templates so they are not pure examples of natural language. We also found that it was too narrow in the range of stereotypes it represented, focusing exclusively on the United States.

C.5.2 Dataset size details

Details on the size of each dataset, including average token counts under the GPT-2 tokenizer, are found in Table C.7.

C.5.3 Prompts for generative models

All data generation was performed with `gpt-3.5-turbo` or `GPT-4`.

Split	Task	# Train	Avg Length Train	# Eval	Avg Length Eval
Val	Country	119	9.58	582	111.47
	Company	86	11.07	421	36.52
	Gender	20	4.25	320	11.69
	Hard Syntax	240	5.44	360	8.54
	Stereoset	1053	8.64	1053	7.89
	Temporal	75	137.37	452	87.86
Test	Country	119	9.74	583	109.61
	Company	86	11.60	403	36.70
	Gender	20	4.40	360	10.73
	Hard Syntax	240	5.38	360	8.54
	Stereoset	1053	8.64	1053	8.02
	Temporal	76	137.42	486	99.67

Table C.7: Number of examples, and average token counts, in the train and evaluation splits of our datasets.

Generalization set E

Country Generating the canonical example statements of country-capital cities (to get some extra fluency in edge cases.)

```
Please generate a statement that the capital of {} is {}.Be fluent,
adding or removing 'the' as necessary. Generate it as a python
string, with absolutely no other markup or commentary.
```

Generating paragraphs eliciting the capital of the country:

```
Please generate a varied, interesting paragraph that (1)
first mentions the name of the country in the sentence below,
and then (2) later, brings up the idea of the country's capital,
and then (3) says the name of the capital. It should be natural,
but rather clear that the capital is about to be mentioned. Here
is the statement from which to pull the capital and country: {}.
```

we generate five such paragraphs in the same context; after each one, all previous paragraphs are conditioned on, along with the following intermediary prompt:

```
Great; please generate another one with varied structure,
ensuring that the prefix before the first time that the capital
is mentioned clearly indicates that the capital is about to
be mentioned.
```

Company For generating a paragraph about company-CEO relationship:

```
Please generate a varied, interesting paragraph that (1) first mentions
the name of the company in the sentence below, and then (2) later,
brings up the idea of the company's CEO, and then (3) says the name
of the CEO. It should be natural, but rather clear that the CEO is
about to be mentioned. Here is the statement from which to pull the
CEO and company: [country]
```

we generate five such paragraphs in the same context; after each one, all previous paragraphs are conditioned on, along with the following intermediary prompt:

```
Great; please generate another one with varied structure, ensuring
that the prefix before the first time that the CEO is mentioned
clearly indicates that the CEO is about to be mentioned.
```

Gender Bias We paraphrased some of the evaluation prompts of (Hewitt et al., 2023) with the following:

```
Please generate a short paraphrase of this fragment. It's critical
that the paraphrase be continuable by a pronoun like 'he', 'she',
or 'they'. It's also critical that the [career] token is maintained
identically. Do not use a pronoun in the prefix. Be creative.
Here's the prefix: '{}'
```

Stereoset Not used.

Hard Syntax To generate a semantically coherent disambiguating sentence from a prefix:

```
Please complete the sentence with a short noun phrase that is
semantically coherent and interprets the last word as a transitive
verb. Ensure the transitive verb is not part of a multi-verb phrase.
The noun phrase should be the object of the verb. At most 6 words.
Only generate the completion; do not generate the whole input
sentence. The verb is {}; make sure it's interpreted as a verb
in the sentence.
```

Temporal To generate a short description of an entity:

```
lease generate a varied, interesting paragraph that (1) first mentions
the name of the person/company/entity/idea/concept mentioned below,
and then (2) discusses the concept and things relevant to it in a
```

```
short paragraph. It should be natural, informational, factual.
Here is the relevant entity: {}.\n\nNow, generate just your resulting
paragraph, with no additional discussion.
```

Hard negative set H

Country

```
A well known city in {country} is {other_city}.
Here's a fact about it: {fact}
Please generate a varied, interesting sentence that
(1) first mentions the name of the country and then
(2) mentions the fact about the aforementioned city
in the same sentence. However, it's extremely
important that the fact be mentioned before the city
name {other_city} is mentioned, and it should be
natural, but rather clear that the city {other_city}
is about to be mentioned. Generate only the sentence
and nothing else. The provided fact might mention the
capital city of the country in addition to {other_city},
but you should mention {other_city} only.

For example, for Afghanistan's city Herat, here is a fact about
it: Herat is home to the Great Mosque of Herat (Jama Masjid),
a grand example of Islamic architecture from the Timurid period.
An example output is:
Afghanistan boasts Islamic architecture from the Timurid period.
A grand example is the Great Mosque of Herat (Jama Masjid), located
in the city of Herat.

Note how the fact about Herat, i.e. the the Great Mosque, is
mentioned before the city of Herat is mentioned in the same
sentence. You should make sure your sentence has the same
structure.
```

As a heuristic validation:

```
The capital of {country} is {capital}. Using the output format
below, generate a well known fact about a well known city in
this country that is NOT the capital. This fact should be true
only of this other city, and not true of the capital city.
```

```
Examples are landmarks in this other city or historical events
that happened in this city. Explictly think about what is not
true of the capital city {capital} but true of this other
city in {country}
```

Company Same as evaluation set, with different entities.

Gender Bias To generate a story about a person who explicitly uses a set of pronouns:

```
Please write an interesting and relatively short sentence about
a {job} who uses the pronouns "{pronouns}". A pronoun should
appear at least once, but not at the beginning of the sentence.
Explicitly mention the person is a {job}. Stay away from
stereotypes about people who use the pronouns {pronouns}.
```

Stereoset For words/phrases not found in the dictionary, we elicited a short definition with the following:

```
Please generate a short definition for this word. If there’s
a typo, figure out what the word should be but don’t mention it.
The word is {}. Do not add any words like ’the definition of...
is’; instead just write the definition; e.g., for ’manager’,
’someone who controls resources and expenditures’.
Do not titlecase the first word
```

Hard Syntax To generate a semantically coherent sentence with a given subject to test whether the verbs in the canonical examples can still also be used as nouns:

```
Please generate a short, semantically coherent sentence with
the following subject: {}
```

and similarly for the nouns that showed up in the canonical example set:

```
Please generate a short, semantically coherent sentence with
the following word: {}
```

Temporal Same as evaluation set, with different entities.

Ball B_ϵ	Method	Task	LR	KL Penalty	Sense #	Sense Reg.	LoRA Rank	LoRA Layers
0.0001	full	company	5.24e-07	0.108	-	-	-	-
0.0001	lora	company	0.000362	0.139	-	-	171	1-11
0.0001	senses	company	0.0155	0.161	9	1000	-	-
0.001	full	company	1.04e-05	0.263	-	-	-	-
0.001	lora	company	0.00344	0.102	-	-	155	5-7
0.001	senses	company	0.0304	0.1	10	1000	-	-
1e-05	full	company	2.54e-07	0.196	-	-	-	-
1e-05	lora	company	0.000362	0.139	-	-	171	1-11
1e-05	senses	company	0.00312	0.443	10	1000	-	-
0.0001	full	country	5.73e-06	0.296	-	-	-	-
0.0001	lora	country	0.000764	0.275	-	-	184	1-11
0.0001	senses	country	0.0149	0.421	8	1745	-	-
0.001	full	country	6.46e-06	0.352	-	-	-	-
0.001	lora	country	0.00244	0.118	-	-	69	1-12
0.001	senses	country	0.0149	0.421	8	1745	-	-
1e-05	full	country	2.72e-06	0.636	-	-	-	-
1e-05	lora	country	0.000764	0.275	-	-	184	1-11
1e-05	senses	country	0.00138	0.109	11	159	-	-
0.0001	full	gender	2.54e-07	0.196	-	-	-	-
0.0001	lora	gender	0.00228	0.149	-	-	8	3-9
0.0001	senses	gender	0.0201	0.385	8	1000	-	-
0.001	full	gender	1.04e-05	0.263	-	-	-	-
0.001	lora	gender	0.000424	0.515	-	-	129	3-9
0.001	senses	gender	0.0201	0.385	8	1000	-	-
1e-05	full	gender	2.54e-07	0.196	-	-	-	-
1e-05	lora	gender	0.000103	0.469	-	-	211	3-10
1e-05	senses	gender	0.0201	0.385	8	1000	-	-
0.0001	full	stereoset	8.43e-09	0.839	-	-	-	-
0.0001	lora	stereoset	0.000103	0.469	-	-	211	3-10
0.0001	senses	stereoset	0.00457	0.151	5	1000	-	-
0.001	full	stereoset	4.23e-08	0.395	-	-	-	-
0.001	lora	stereoset	3.02e-05	0.559	-	-	19	3-10
0.001	senses	stereoset	0.00558	0.301	6	1000	-	-
1e-05	full	stereoset	5.17e-09	0.373	-	-	-	-
1e-05	lora	stereoset	4.07e-05	0.606	-	-	144	4-8
1e-05	senses	stereoset	0.000743	0.749	9	1000	-	-
0.0001	full	temporal	4.2e-06	0.107	-	-	-	-
0.0001	lora	temporal	0.00153	0.456	-	-	53	2-11
0.0001	senses	temporal	0.0149	0.169	11	1000	-	-
0.001	full	temporal	4.2e-06	0.107	-	-	-	-
0.001	lora	temporal	0.00153	0.456	-	-	53	2-11
0.001	senses	temporal	0.0149	0.169	11	1000	-	-
1e-05	full	temporal	4.2e-06	0.107	-	-	-	-
1e-05	lora	temporal	0.00274	0.266	-	-	154	3-9
1e-05	senses	temporal	0.00773	0.11	5	1000	-	-
0.0001	full	syntax	4.23e-08	0.395	-	-	-	-
0.0001	lora	syntax	6.29e-05	0.785	-	-	184	5-7
0.0001	senses	syntax	0.00235	0.368	10	1000	-	-
0.001	full	syntax	5.24e-07	0.108	-	-	-	-
0.001	lora	syntax	0.000103	0.469	-	-	211	3-10
0.001	senses	syntax	0.00235	0.368	10	1000	-	-
1e-05	full	syntax	1.1e-08	0.78	-	-	-	-
1e-05	lora	syntax	4.99e-07	0.727	-	-	69	4-9
1e-05	senses	syntax	0.00235	0.368	10	1000	-	-

Table C.8: Best hyperparameter for each degradation ball-method-task combination for the Backpack language model.