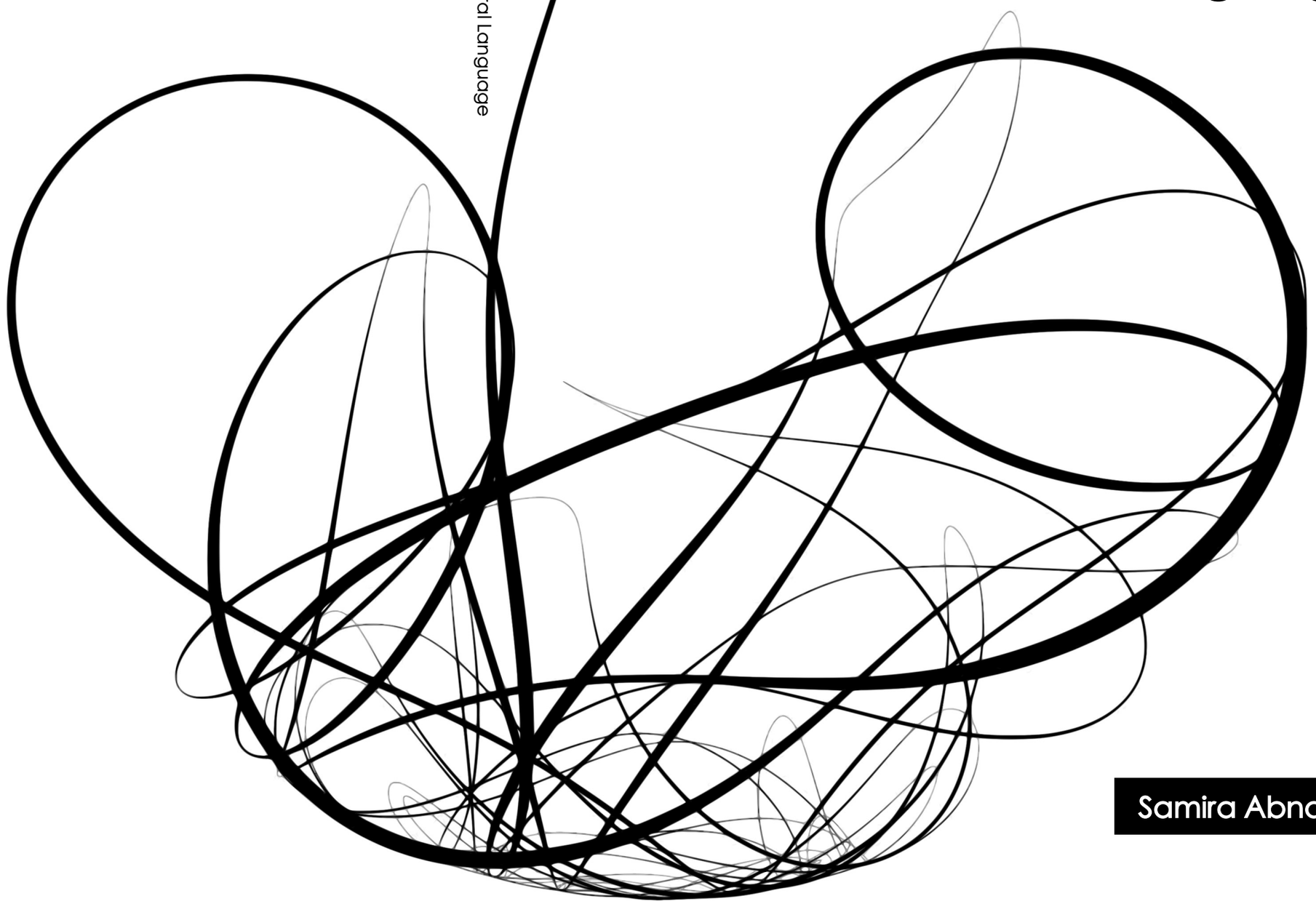
Inductive Biases for Learning Natural Language

Inductive Biases for Learning Natural Language



Samira Abnar

Inductive Biases for Learning Natural Language

Samira Abnar

Inductive Biases for Learning Natural Language

ILLC Dissertation Series DS-2023-04

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

The research for/publication of this doctoral thesis received financial assistance from the Language in Interaction consortium (LiI), which is subsidized by the Netherlands Organization for Scientific Research (NWO).

Cover design by Mostafa Dehghani. The digital art used on the cover is the path of a double pendulum traced out with lyrical, calligraphic lines created by Nathan Selikoff. Printed and bound by Ipskamp Printing.

ISBN: 978-94-6473-081-4

Inductive Biases for Learning Natural Language

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op maandag 24 april 2023, te 14.00 uur

door Samira Abnar
geboren te Tehran

Promotiecommissie

Promotor:	dr. W.H. Zuidema	Universiteit van Amsterdam
Copromotor:	dr. W. Ferreira Aziz	Universiteit van Amsterdam
Overige leden:	dr. A. Alishahi	Tilburg University
	dr. S.L. Frank	Radboud Universiteit
	prof. dr. R. Fernández Rovira	Universiteit van Amsterdam
	dr. ir. J. Kamps	Universiteit van Amsterdam
	prof. dr. F. Roelofsen	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

to my fellow beings
in Iran, Afghanistan, and all over the world
who are honorably fighting for their right to pursue science

Contents

III Effects of Inductive Biases

IV Conclusion

Acknowledgments

I can't believe this is all done! If there is one thing in this world that I have always struggled with, it is finishing the projects that I start. This journey has been full of ups and downs, and doubts, more than anything else. Without the guidance, help, and support from everyone, I would not have been able to start this, let alone cross the finish line. For me, this book is a celebration of the years I have spent at ILLC as a PhD student. I am so grateful for all the things I have learned, all the people I have met, and all the friends I have made during this time. It's funny how my name is the only one on the cover of this when the credit should go to so many of you who are probably reading this now. If you are reading this, you know how much I owe this moment to you. Without further ado, Thank you!

Dear Jelle, thank you for giving me the opportunity to work with you at CLCLab. Thank you for guiding and supporting me along the way. Thank you for all the inspiring discussions we had and for always making me feel like I knew what I was talking about, even when I was just blabbering.

Dear Wilker, thank you for being my co-supervisor. Thank you for all the great advice and tips. Thank you for all the thoughtful and elaborate explanations of scientific concepts during our reading groups and over lunch breaks. Thank you for reviewing my papers on very short notice. I wish I had more time to work with you and learn from you.

Dear ILLC office, thank you for all your help and support. Thank you for always being there and assisting me with all my troubles and requests. Thank you for making everything possible with your support.

Dear committee, thank you for taking the time to read my thesis and for accompanying me during the final stage of defending my PhD.

Dear Mom and Dad, I have given a lot of thought to finding a way to thank you beyond clichés. I just couldn't figure it out. I am not mentioning you here just because everyone is supposed to do so. You are the only lucky card I have had in life. You are the beginning of all the good things I have encountered in life. You are the start of all the opportunities I have had. Thank you a million times.

Dear Mostafa! Thank you for being such a great family, friend and collaborator to me. Thanks for listening to my crazy ideas and understanding them. Thanks for being and having everything I am not and don't have and for sharing that with me. Thank you for

never letting me give up and for always being there for me.

Dear Afra and Ali, I put together most of this thesis when I was visiting you. Several people had suggested that I retreat to a peaceful cabin in a jungle to focus on my writing, but little did they know there is no place more peaceful and calming for me than where you are. Thank you!

Dear family, friends, colleagues, and former supervisors and teachers, I am hoping I will meet you in person sometime soon and I can hug you and tell you how much I appreciate you and how much your presence alone has helped me over these years. You all deserve more than individual thank-you notes to remind you how amazing and inspiring you are. I can not begin to mention your names here as that is going to take a whole other book that deserves to be written maybe at another time. I feel super privileged, to know you all and have you by my side.

I keep asking myself, what have I done really to deserve so much in this chaotic world we live in? or what can I do to be worthy of all this? I can just hope that the knowledge and skills I have gained will allow me to contribute to solving some of our problems in this world. May we have a world, more fair, more happy, more beautiful.

Samira Abnar
Winter 2023

1 Introduction

1.1 Main Goal and Motivation

Machine Learning and Natural Language Processing (NLP) research have often been inspired by our understanding of human cognition and intelligence. For a long time, the state-of-the-art NLP models were based on symbolic pipelines inspired by cognitive linguistics theories of how syntax and semantics interact. Most recently, Neural Networks (NNs), machine learning models loosely inspired by the structure of the brain, are achieving remarkable performance on a variety of tasks including understanding and generating natural language.

Interestingly, ideas flow between cognitive science and machine learning in both directions. As our knowledge of human cognition is still very limited, in many cases, computational simulations of how machine learning models learn and infer different tasks are used to provide insight into human cognition and unlock mysteries about our intelligence [Güçlü and van Gerven, 2015, Huth et al., 2012, Mitchell et al., 2008, Rumelhart and McClelland, 1986, Zhuang et al., 2021]. This is a loop, where we try to build better machine learning models based on our understanding of how the human brain works, and ultimately, these models feedback into cognitive models.

One of the most interesting aspects of human intelligence is perhaps their ability to learn and process natural language. The big question that has been around for decades is: what is special about the human brain that enables having such a complicated communication system and transferring it through generations? How can we design and train machine learning models that can learn language as efficiently and effectively as humans?

One of the main topics of debate in the field of language acquisition has been if the

training signals children receive are enough, qualitatively and quantitatively, to learn language if they have no prior (innate) knowledge about it. Tomasello [2009] suggests that the language input children receive is enough if language is considered as "a set of symbolic instruments for directing the intentional and mental states of others", and general cognitive abilities of humans such as categorization, analogy, statistical learning, etc, are taken into account.

These cognitive abilities (or constraints) are generally referred to as learning biases and can be interpreted as our prior knowledge or assumptions about how the world works. Without any learning biases, we would not be able to generalize to deal with examples beyond what we have been exposed to.

There are different ways of describing learning biases: (a) The description can be in terms of a set of constraints and desires, mental or physical, that determine the optimal way of learning a skill. For instance, the human attention mechanism seems to be a sequential process, i.e., we can not simultaneously attend to multiple sensory inputs . Hence it makes sense for the human communication system, language, to evolve to be based on sequential rules and structures. (b) It can be described in terms of the resulting behaviour, e.g. humans have learning biases that make it easy for them to learn compositional rules and patterns. (c) It can be described in terms of a set of assumptions about how the world works. e.g., assuming that we live in a stable world and words and their meanings do not change frequently.

There have been many efforts trying to identify and understand the nature of the learning biases that enable language learning in humans. These biases could be domain-general or domain-specific. For example, an interesting learning bias, described at the implementation level is the maturing memory bias [Elman, 1993], i.e., the children's working memory is very limited and its capacity increases as it matures [Baddeley, 1992], this biases children to learn simpler structures earlier. Hence, natural languages have evolved to be easily learnable by a learning algorithm that starts small and grows gradually [Deacon, 1997].

Other examples of learning biases for language, described at the behavioural level, are biases towards certain syntactic universals such as word order universals [Culbertson et al., 2012, Greenberg, 1963], and word learning biases [Markman, 1990], such as whole object assumption [Markman, 1991], taxonomic assumption [Markman, 1991], mutual exclusivity assumption [Markman, 1991, Merriman et al., 1989], noun-category bias [Waxman and Kosowski, 1990], and shape bias [Landau et al., 1988]. A more general form of learning bias that is shown to play a major role in the language acquisition process and the evolution of natural languages is the regularization bias [Culbertson et al., 2012, Hudson Kam and Newport, 2005, Marcus et al., 1992, Singleton and Newport, 2004].

The parallel to the concept of learning biases in humans, in machine learning is the concept of inductive biases. Inductive biases are data-independent factors that enable and determine the generalization behaviour of the models beyond the training data [Mitchell,

1980]. In this thesis, our main goal is to study the inductive biases of different neural network-based language models and their connection to underlying processes in the human brain. More specifically, we propose different techniques that allow us to evaluate, investigate and compare solutions learned by different models to reveal the impact of different choices in designing neural network models for language processing.

Our focus is on two families of models that have been successful in achieving state-of-the-art results on a wide range of NLP tasks in the past few years: (1) LSTMs: variants of RNNs with a gating mechanism, (2) Transformers: NNs consisting of layers of self/cross attention stacked on top of each other.

The motivation for us to focus on inductive biases is two folds; first, the literature on cognitive linguistics which collectively suggests that there is a set of learning biases that enable humans to learn and process language; second, machine learning literature about inductive biases, that suggest a reliable out-of-distribution generalization can be achieved by injecting some prior knowledge about the target distribution into learning algorithms.

To build machines that learn language efficiently and can generalize beyond the distribution of their training data in a similar manner humans do, on the one hand, we need to identify and confirm the main learning biases that enable humans to learn the natural language, and on the other hand, we need to find a way to incorporate them as inductive biases in machine learning algorithms. In this thesis, we take a small step toward this goal by introducing methods and designing experiments that can illustrate the impact of inductive biases regarding the performance of the models as well as their similarity to how the human brain works.

In the rest of this chapter, first, we discuss inductive biases in machine learning, §1.2. We provide an overview of the concept of inductive bias in machine learning and discuss the challenges in quantifying, identifying and injecting inductive biases in machine learning algorithms.

Next, we discuss how using different techniques to compare the human brain, and neural networks, two systems that are hard to probe and explain (often referred to as black boxes), can lead to interesting insights and deepen our understanding of how these black boxes operate, §1.3. Finally, we provide an overview of the structure of the thesis and highlight the main contributions in each part, §1.4.

1.2 Inductive Biases in Machines

Let's step back and revisit the concept of learning in general. A simple and naive learning strategy is to memorize the environment and the experiences. However, simply memorizing past experiences, without any generalization, without any loss of details, often would not be useful as past experiences repeated with the exact details are rare. For a learner to acquire knowledge beyond memorizing the experience, i.e., the

environmental interactions, the learner has to be equipped with a mechanism that allows it to generalize beyond what it has been exposed to.

This mechanism can be in form of prior knowledge about the environment or a systematic way to prioritize certain generalizations if supported by the evidence, or it can be rooted in the underlying mechanism of the learner, the learning algorithm and its capacity and limitations. In machine learning the set of assumptions and prior knowledge incorporated into the learning algorithm is referred to as inductive biases.

Inductive biases are the characteristics of learning algorithms that influence their generalization behaviour, independent of data. They are one of the main driving forces to push learning algorithms toward particular solutions [Mitchell, 1980]. Having the right inductive biases is particularly important for obtaining high performance when data or computing resources are a limiting factor, or when training data is not perfectly representative of the conditions at test time, i.e., when we expect the models to generalize to out-of-distribution data.

In the absence of strong inductive biases, a model can be equally attracted to several local minima on the loss surface; and the converged solution can be arbitrary and affected by random variations in, for instance, the initial state or the order of training examples [Dodge et al., 2020, Sutskever et al., 2013].

There are two types of inductive biases: restricted hypothesis space bias and preference bias. Restricted hypothesis space bias determines the expressiveness of a model, i.e., certain solutions are not learnable by the learning algorithm at all, while preference bias weighs the solutions within the hypothesis space [Craven, 1996]. In the case of the latter, all solutions are learnable if supported by evidence.

While injecting strong inductive biases into learning algorithms might seem unappealing as it might restrict the expressivity of the models, and difficult, as it requires prior knowledge about the desired generalization behaviour, it is impossible for an algorithm without any inductive biases to generalize beyond its training data. For learning algorithms, to consistently and reliably generalize to both in and out of distribution data, they need to have proper inductive biases.

Some examples of the classes of inductive biases as introduced by Mitchell [1980] are: (1) Factual knowledge of the domain, e.g., rules of symmetry or compositionality in the data. (2) Intended use of the learned generalizations, e.g., bias toward making less false negative examples. (3) Bias toward simplicity, e.g., regularization techniques. (4) Analogy with previously learned generalizations. (5) Knowledge about the source of training data, e.g., if the order of training samples obeys a specific curriculum strategy.

When studying the inductive biases of machine learning algorithms we are often faced with two questions:

- How can we quantify the inductive biases of a learning algorithm?
- What are the sources of inductive biases? Having a certain prior knowledge about

the scope of the solution how can we inject this knowledge in form of an inductive bias into a learning algorithm?

Finding the answers to these questions may eventually lead to discovering a set of minimal essential inductive biases to enable the learning algorithms, efficiently and effectively.

1.2.1 Quantifying Inductive Biases

Identifying and quantifying the inductive biases of a model is a challenging problem. Based on the definition of inductive bias, i.e., what determines the generalization behaviour of the model independent of training data, to reveal the inductive biases of a learning algorithm, we need to investigate its generalization behaviour on a variety of tasks and data distributions. Profiling the performance of the model on various metrics/tasks that are independent of the objective/task that the model is trained on can reveal much about the inductive biases of the models. For example, if a language model can consistently generalize to inputs of varying lengths that are unseen during training, it shows that it has an inductive bias toward learning compositional rules that allow length generalization.

Furthermore, based on the bias-variance trade-off, to measure the strength of the inductive biases of a model, independent of the type of inductive bias, we can track the variance of the performance of the model with respect to their performance on both in-distribution and out-of-distribution tasks, when the model is trained on different parts of the training data. The lower variance means a stronger inductive bias. More generally and intuitively, the variance of a model in terms of its performance, the errors it makes, and its representational stability as a result of any source of stochasticity (initialization, order of training data, etc), as well as its sample efficiency could potentially be indicators of the strength of the inductive biases of the model.

1.2.2 Sources of Inductive Biases

Any learning algorithm that can generate outputs for unseen inputs has some sort of an inductive bias that allows it to generalize to unseen inputs regardless of the fact that its generalization behaviour is what we expect/desire or not [Micheli et al., 2020]. Inductive biases can be rooted in different components of the learning algorithm, e.g., pre-processing steps and input representations, specific parametrization of the learning algorithm, initialization and training strategy, the training objective and the optimization algorithm. For instance, applying different forms of regularization by adding noise to the data or adding auxiliary losses to put extra data-independent constraints on the parameters of the models are one the most common ways of biasing the hypothesis space of the models toward simpler solutions. For Bayesian models, the assumptions

about the prior distribution, are an explicit source of the inductive biases. In the case of neural networks, network architecture, the connectivity pattern between the parameters of the network, and different forms of parameter sharing are among different sources of their inductive biases.

We can describe inductive biases in terms of their implementation or their effect. Knowing the desired generalization behaviour, the description of the inductive bias in terms of its effect, we might be able to design learning algorithms to restrictively or preferably reflect that. Bias toward simplicity is an example of this. Another example is when we do have a piece of prior knowledge about certain rules of symmetry in the data, and we select neural networks with specific connectivity patterns that are equivariant or invariant to specific changes in the input. For instance, specific convolution neural network architectures could be translation invariant, regardless of how they are trained.

On the other hand, intuitively, every implementation detail of a learning algorithm can potentially affect its hypothesis space and how it explores it. Different architectural choices in designing neural networks such as the connectivity patterns or size and shape of the network can impact their sample efficiency and their generalization behaviour. For example, parameter sharing in space/time, e.g., convolutional neural networks [LeCun and Bengio, 1998], recurrent neural networks [Elman, 1990], and Transformers [Vaswani et al., 2017], or in depth, e.g., universal transformers [Dehghani et al., 2019]. While, we assume these kinds of constraints inject some form of inductive bias into the models, in many cases we do not know the exact nature of these biases beyond speculations. Hence, often we explain these inductive biases in terms of their implementation rather than their effect. It is noteworthy that while any detail in the design and implementation of learning algorithms can be a source of inductive bias, discussing these impacts in terms of inductive biases rather than the details specific to each model, provides a more unifying view of different learning algorithms.

1.3 BlackBox Meets BlackBox

Today's neural network models of language are impressively good in learning representations that can be used to successfully solve various linguistic tasks [Brown et al., 2020, Devlin et al., 2019, Radford et al., 2019]. What do these models learn about language and how do they learn it? There is an active line of research dedicated to answering these questions [Brown et al., 2020, Giulianelli et al., 2018, Hupkes et al., 2018, Kim et al., 2019, Linzen et al., 2016, Marvin and Linzen, 2018, McCoy et al., 2019, Tenney et al., 2019a, van Schijndel et al., 2019].

These efforts are in parallel to the efforts in cognitive linguistics to try to understand language processing in the human brain. Studying brain imaging data to confirm cognitive linguistic theories and to understand human capabilities and biases for learning language is the other side of this coin which is extensively explored [Caramazza and

Zurif, 1976, Damasio et al., 1996, Devlin et al., 2004, Ettinger et al., 2014, Friederici, 2002, Hagoort, 2005, Patel, 2003, Solomyak and Marantz, 2010, Ullman et al., 1997, Wang et al., 2003].

Inevitably, these two lines of research, understanding how neural networks learn and process language and how language is processed in the human brain, have crossed. Besides, exchanging probing techniques and processes across these two worlds, deep learning and cognitive science, we can use neural network language models to shed light on how the human brain processes language and use brain signals and our understanding of how the human brain works to evaluate neural language models.

Research on using computational simulations for providing evidence to validate or invalidate various language acquisition or evolution theories with symbolic or connectionist approaches is not new [Alhama and Zuidema, 2019, Frank and Tenenbaum, 2011, Hausser and Hausser, 2001, Plunkett and Marchman, 1991, Wintner, 2010].

Neural networks, i.e., connectionist approaches, provide a nice test bed for evaluating theories of cognition. In contrast to symbolic models, with neural networks, we have more degrees of freedom for the type and strength of the inductive biases. With symbolic models, we need to have some prior knowledge and assumptions about how the task at hand needs to be solved. This is not necessarily the case for neural networks, and we can apply weaker notions of inductive biases rather than dictating the solution to the model. E.g., instead of having a model that can only learn hierarchical solutions, we can have a model that has a bias toward learning hierarchical solutions. The downside of using neural networks for the computational modelling of cognitive phenomena is their interpretability challenge, as the characteristic of the solutions these models converge to are intractable. On the plus side, understanding the underlying mechanisms of these models is way easier than probing the human brain.

To close the loop and bridge the efforts in neuro-linguistics and computational linguistics, we need computational modelling frameworks that not only can account for the abilities of humans to learn language at the behavioural level but can also explain the neurological signals from the human brain when learning and processing language. For example, we can build neural networks that learn to process language and use the internals of these models to predict the brain signals of human subjects, while they are performing a similar task. While this has been the focus of many recent research [Alishahi et al., 2020, Beinborn et al., 2019, Caucheteux and King, 2021, Chehab et al., 2021, Gauthier and Ivanova, 2018, Heilbron et al., 2021, Jain and Huth, 2018, Mitchell et al., 2008, Murphy et al., 2012, 2018, Schrimpf et al., 2020, Schwartz and Mitchell, 2019, Sun et al., 2019a, Toneva and Wehbe, 2019], there are still many questions and unresolved challenges that require our attention.

One thing that is often missing in these studies is ensuring that improvements in the performance of the models or their behavioural similarity with human subjects at different levels are causally correlated with their capacity to explain brain signals. Another major issue, especially when using neural network models, is our lack of

understanding of what is captured in the representation of the models as well as in the neurological signals. Here, the question is how do we probe one black box with another one? Can we learn any new insights about how language is learned and processed in neural network language models or the human brain, solely by comparing black boxes? And how can the efforts for the interpretation of neural networks fill in the gaps? We believe exploiting and extending the interpretability toolkit for better understanding the underlying mechanisms and processes implemented by neural network language models is an essential part of the efforts for better understanding how language is implemented in the human brain through the lens of neural networks.

1.4 Overview of the Thesis

In this thesis, we follow prior work on using neural networks to understand inductive biases that are necessary or helpful for learning a natural language. For this purpose, we investigate neural network models with different architectures trained under different settings and with different language modelling related objectives.

We ask: what are the inductive biases that are useful, or necessary for learning to process language and whether they are connected to human cognitive processes and learning biases? To address the aforementioned challenges we aim to first understand the effect of different sources of inductive biases. Different architectural choices in designing neural networks along with the training algorithms and objective functions provide different kinds of inductive biases that affect different qualitative factors of the solutions these models converge to.

By investigating the effects of these factors and comparing the solutions under different conditions to each other and also to signals from human references, we can investigate the connection between different sources of inductive biases in neural networks and the learning biases of humans. If we find a specific set of inductive biases that significantly push the behaviours of the models and the characteristics of the solutions they learn toward human language processing behaviour, we can use this as a shred of additional evidence to support theories about the importance of those biases in the language learning process.

Our focus, in this thesis, is exploring different techniques that allow us not only to investigate the solutions neural network language models converge to but also how we can make a bridge between the human brain and the computational models beyond the performance of models and humans on different tasks.

1.4.1 Part I: Interpretation Techniques for Language Models

In part I, we employ and propose different techniques to study the characteristics of different language models and the impact of different factors in the solution they

converge to.

In chapter 2, we propose to use the representational similarity analysis framework to study representations obtained from LSTM, non-autoregressive and autoregressive Transformer based language models. We investigate how factors such as context length, the architecture of the network, and training objective impact the representational spaces of these models and how they evolve across layers. We argue that understanding how the representational spaces evolve and change can shed light on the internal mechanisms of these models.

In chapter 3, we focus on techniques specific to attention-based models, i.e., Transformers. A straightforward approach for analyzing attention-based models is to investigate the attention patterns in different layers of these models. In some cases, these attention scores are treated as a proxy of the contribution of input tokens in the output of these models. The major issue with these types of analysis is that they do not take the evolution of token representations across layers of attention block into account. I.e., the representations in the last layer of the model do not necessarily carry any information about their corresponding input token. In our work, we try to understand how attention scores in Transformer based models can be used to explain the characteristics of their solutions by taking the propagation and mixing of information across the layers of these models into account.

1.4.2 Part II: Models and Brain

In parallel to our efforts to understand the effect of different factors on the solutions different neural language models converge to, in part II, we make an effort to build on top of existing approaches that use brain activity patterns as a biological reference to evaluate language models.

We start our journey, in chapter 4 by using regression techniques to predict and decode brain activity patterns to evaluate the similarity of representations obtained from different word embedding models for nouns with brain signals. We find that general-purpose word embedding models such as variants of word2vec are more correlated with brain signals than hand-crafted word feature vectors that are specifically designed for explaining brain signals collected in a particular study.

In chapter 5, we extend our efforts to more complicated settings where words are provided in a context of a story to both humans and machines. Our ultimate goal is to investigate the connection between the inductive biases of neural language models and cognitive processes involved in language learning and understanding. We find that, among existing neural network architectures, recurrence has a significant role in facilitating learning structures needed to solve language tasks more similar to the human brain.

1.4.3 Part III: Closer Look at The Effects of Inductive Biases

In part III, we look more directly into the effect of inductive biases of models. Taking into account our experiments in previous chapters, and considering the literature on the advantages of recurrent neural networks in solving language tasks and their cognitive relevancy, we try to understand the inductive biases of this family of models. Here, by cognitive relevancy we mean (1) how these models, perhaps at a very abstract level, are inspired and can be mapped to some cognitive or neural processes in the human brain; and (2) how similar/aligned are the representations obtained from these models to signals obtained from the human brain and if they can inform us about the underlying mechanism of language processing in the human brain.

In chapter 6 we break down the roots of the recurrent inductive bias into three factors of (1) incremental processing of input, (2) memory bottleneck, and (3) parameter sharing in time, and design experiments to qualitatively show the impact of each in processing language.

Following these experiments, in chapter 7, we aim to use knowledge distillation as a framework to compare the inductive biases of different neural network architectures, and as a technique to empirically study the expressivity of neural networks architectures versus the learnability of a desired generalizable solution for them. In this chapter, we demonstrate that in the knowledge distillation process, having a teacher with proper inductive biases and a student model that is efficient with respect to the teacher, the effects of the inductive biases of the teacher model transfer to the student model.

PART

I

Evaluating Neural Language Models Beyond their Performance

A major focus of recent research on deep learning has been to improve the expressivity of machine learning models such that they are less complex in terms of space and time at training and inference. We have come a long way from multi-layer-perceptrons, to different variants of RNNs [Cho et al., 2014, Hochreiter and Schmidhuber, 1997, Jordan, 1997], CNNs [LeCun and Bengio, 1998], and Transformer models [Vaswani et al., 2017]. However, having an expressive enough model that has the capacity and power to estimate or execute the target functions or processes of interest is just one side of the story.

In parallel to this, there has been a lot of efforts to understand how one can efficiently guide these models toward the target solution and benefit from their expressivity by carefully designing the optimization algorithms and training strategies, baking in the proper inductive biases, and preparing quantitatively and qualitatively sufficient data for training and evaluating these models [Devlin et al., 2019, Dosovitskiy et al., 2021, Radford et al., 2021].

The common approach for measuring the progress along these axes is the performance of the models on a set of benchmark datasets in both in-distribution and out-of-distribution settings, and there exist many efforts addressing the need to have unbiased and diverse benchmarks. However, there is always a chance that the models find shortcuts to achieve high performance on given benchmarks without generalizing in an intended manner [Geirhos et al., 2020], and we need proper probing techniques to compare and understand the characteristics of the solutions the models converge to.

The many hyper-parameters that can impact the performance of the models, such as their architecture, regularization techniques, the optimization algorithm and the dataset used to train them, along with their inherent underlying mechanism that does not directly map input features to outputs, e.g., having multiple layers and different sources of non-linearity, have raised many questions and concerns about why, when, and how these models work?

Not only understanding the underlying mechanisms and the successes and failures of existing models can lead us toward more powerful and robust models, but also it is necessary to be able to know when we can rely on these models in practical settings.

Among the approaches for evaluating neural networks beyond their performance on given benchmarks, the most common practices are:

- Diagnostic Classification [Tenney et al., 2019b, Veldhoen et al., 2016, Voita and Titov, 2020]: which tries to understand the underlying mechanisms of the models by revealing the information predictable by different components of the models.
- Comparative analysis of representational spaces of different models [Abnar et al., 2019, Kornblith et al., 2019, Laakso and Cottrell, 2000]: which focuses on how representational spaces of the models and their different component evolve during training, or how different factors impact the representational spaces of the models at different levels.

- Feature attribution methods [Lundberg and Lee, 2017]: which are mainly concerned with identifying which parts of the input are used by the models, at different stages, in order to make certain predictions.
- Instance attribution methods [Koh and Liang, 2017, Pezeshkpour et al., 2021]: which are concerned with how the prediction of the models is influenced by different instances in the training data.
- Profiling the performance of models with respect to a wide range of tasks and metrics, other than metrics that are necessarily correlated with the training objective [Srivastava et al., 2022]. Or exploring the performance of models through different ways of interaction mechanisms with them. E.g., different ways of priming or prompting [Liu et al., 2022, Sinclair et al., 2022, Wei et al., 2022].

Our main goal here is to explore different interpretation techniques to understand how different choices of network architecture and training objectives for models of language impact the solution they converge to. Different network architectures and different training objectives can impose different inductive biases on models, which could lead the models toward different solutions. We are interested in the cognitive relevancy of different design choices in building language models, and here we take the primary step to recognize factors with non-trivial effects on the final solution the models converge to.

In chapter 2, we propose to use representational similarity analysis to obtain a better understanding of the impact of model architecture and training objective on the final solution by comparing representational spaces of different models and different components of the same model. In chapter 3, we propose simple techniques that can improve the interpretability of attention weights in self-attention-based models, which can then be used to quantify the information flow in stacks of self-attention layers and uncover the internal processes of these models.

2

Analyzing Representational Spaces

Representational similarity analysis allows us to compare heterogeneous representational spaces [Laakso and Cottrell, 2000]. The key idea is simple: instead of directly trying to map the dimensions of the representational spaces, we measure the relational similarity between them, by first constructing a similarity/co-variance matrix for each model. This approach of comparing representational spaces has two advantages: (1) It treats representational spaces as blackbox; it does not need to know how a model represents objects, words or sentences, but only how similar those representations are to each other; (2) It is invariant to trivial general changes in the spaces, e.g., when everything shifts or scales.

In this chapter, we propose to use representational similarity analysis for understanding the importance and effect of different factors in the solution the models converge to, by measuring the sensitivity of the representational spaces to isolated changes. We call

This chapter is an extension of primary experiments presented in the following paper (most of the content is not published before).

- Samira Abnar, Lisa Beinborn, Rochelle Choenni, and Willem Zuidema. 2019. Blackbox Meets Blackbox: Representational Similarity & Stability Analysis of Neural Language Models and Brains. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 191–203, Florence, Italy. Association for Computational Linguistics.
- List of contributions is as follows. Samira Abnar: Designing and running the experiments, Writing the paper. Lisa Beinborn: Contributed to the discussion about the paper. Helped in revising the earlier versions of the paper. Rochelle Choenni: Contributed to the discussion about the paper. Helped with some of the visualizations. Willem Zuidema: Guiding the research, Writing the paper.

this approach *Representational stability analysis* (ReStA). While in representational similarity analysis, we compare representations in models, and model components, ReStA compares instances of the *same* model, while systematically varying a single parameter.

2.1 Representational Similarity

Model comparison beyond the final performance of the models on a set of benchmarks can potentially shed light on understanding the cons and pros of different choices made in building and training them as well as understanding their underlying mechanisms.

Diagnostic classification and probing [Conneau et al., 2018, Hupkes et al., 2018, Veldhoen et al., 2016, Voita and Titov, 2020] rely on discrete/symbolic representational spaces, i.e., different ways of classifying/clustering input examples. Moreover, these approaches are applied in a more in-direct setup, e.g., if we want to compare model A with model B, we define a set of rules/properties, and measure independently how much of these rules and properties are captured by each model. In this chapter, we focus on methods that allow us to directly compare the representational spaces of different models, and use these methods to investigate the difference between various language models as well as to understand what kind of information is captured by different components of these models.

How can we gain any insights about the characteristics of a solution a model converges to by looking into its representational space? One approach could be to compare the representational space of the model we want to study with the representational space of other models with strong inductive biases toward specific solutions. The main challenge then would be to identify the hypothetical solutions a priori and have a mechanism to explicitly implement them.

Another approach would be to compare the representational spaces in controlled setups where the roots of their differences are known to us in advance. Often we work with models that have different characteristics, e.g., have different architectures, are trained under different regimes and with different training data, etc. Comparing the representational spaces of models with such differences in controlled settings can lead to new insights about the underlying mechanisms of the models. By measuring how much a certain choice in designing and/or training a model impacts the representational space of the model, we can infer if information relevant to that particular design choice is reflected in the representational space. For example, by studying how the representations evolve across different modules, or during training, we can speculate about the underlying reasoning processes of the models.

Representational Space We can define representational space as the mapping from input examples, $x_i \in X = \{x_0, x_1, ...x_n\}$, to a d dimensional space: $G : X \rightarrow G(X)$.

When the goal is to compare the representational spaces of different components of neural networks, the comparison technique should satisfy two main criteria: 1. Applicable across heterogeneous models, and representations with different dimensionality. 2. Invariant to trivial changes in the representations. As discussed in [Kornblith et al., 2019], given two representational spaces of G and F, we seek metrics that are:

1. Invariant to orthogonal transformations, i.e.,

$$d(G,F) = d(GU,FV), \tag{2.1}$$

   where U and V are full-rank orthonormal matrices.

2. Invariant to isotropic scaling, i.e.,

$$d(G,F) = d(\alpha G, \beta F) \tag{2.2}$$

In this chapter, to compare representational spaces, we rely on a technique which is commonly referred to as RSA (Representational Similarity Analysis) which is based on measuring the similarity/correlation of the relations between examples in the given representational spaces. In part 2, where we compare computational models of language with brain activity patterns, we use linear regression besides RSA.

In this section, we provide an overview of different methods for computing similarity/correlation between two given representational spaces. Methods for comparing representational spaces can be grouped into two main groups of methods based on regression and methods based on the relational similarity of the representations. Below we briefly explain existing approaches in these two categories. Our aim is to provide a complete picture of existing methods for comparing representational spaces. In this chapter, we mainly rely on a technique which is commonly referred to as RSA (Representational Similarity Analysis). In part 2, where we compare computational models of language with brain activity patterns, we use linear regression as well.

2.1.1 Regression

An intuitive way to measure the similarity of two representational spaces is to investigate how well the representations obtained for a set of examples from the two spaces are predictable given one of them. An advantage of regression methods is that they assign weights to feature dimensions, hence can provide us with indicators of the relation between individual feature pairs (each dimension). When using regression-based approaches it is important that we use separate training and test sets when measuring the similarity of different representational spaces. Otherwise, especially when dealing with high-dimensional spaces, it might be trivial to find a mapping between any two feature spaces. To measure linear predictability different metrics are often used:

- Distance between the target and predicted representations (e.g., cosine or euclidean distance).
- Explained variance (fraction of variance explained by the predicted representations):

$$R^2 = 1 - \frac{\sum y_i - \hat{y}_i}{\sum y_i - \bar{y}} \tag{2.3}$$

- A notion of accuracy based on the nearest neighbor concept (e.g., the accuracy metrics used by Mitchell et al. [2008] and Wehbe et al. [2014a]). E.g., given two pairs of examples, if the predicted representation of each example is closer to its ground truth representations than the ground truth representation of the other example, they count it as a correct prediction and incorrect otherwise.

Multivariate Linear Regression We can use linear regression or Gaussian process regression with different kernels to quantify the similarity between two representational spaces, G and F.

If F is predictable from G, $d(W_{g\to f}G + B_{g\to f}, F) < \epsilon$, and G is predictable from F, $d(W_{f\to g}F + B_{f\to g}, F) < \epsilon$, we can argue that F and G contain similar information.

Canonical Correlation Analysis Canonical Correlation Analysis (CCA) is a multivariant statistical analysis method to measure the linear association between two sets of random variables. Given n samples from a dataset, $X = x_1, x_2, ...x_n$, to take into account the relation and dependence between the variables in each representational space, CCA quantifies the similarity between $F(X)$ and $G(X)$ by considering canonical variables, a set of variables u and v that are linear combinations of variables of $F(X)$ and $G(X)$, respectively.

$$\begin{aligned} u_i &= a_i F = [a_{i_1} f_1 \; a_{i_2} f_2 \; ... \; a_{i_{|F|}} f_{|F|}] \\ v_i &= b_i G = [b_{i_1} g_1 \; b_{i_2} g_2 \; ... \; b_{i_{|F|}} g_{|F|}] \end{aligned} \tag{2.4}$$

In the above equation, u_i and v_i are the ith canonical variables and a_i and b_i are each the set of coefficient factors for the features of F and G respectively, where $|a_i| = |F|$ and $|b_i| = |G|$. The goal is to find a and b such that given X, u_1 and v_1 have maximum correlation coefficient (ρ_i), for $j > i$, $\rho_j < \rho_i$ and for $i \neq j$, u_i and v_i are uncorrelated. Intuitively, this means linearly projecting each representational space such that the correlation between them is maximized and keeps doing this as many times as the projections of each space remain orthogonal, i.e., rows of U and V are orthogonal. That would give us $min(rank(F), rank(G))$ correlation coefficients, i.e., $|U| = |V| = min(rank(F), rank(G))$. Often the mean correlation coefficient is used as a measure of the similarity of F and G. This way of computing similarity between two representational spaces is invariant to invertible linear transformations.

To compute this we can take the singular value decomposition of $\Sigma_{ff}^{-1/2}\Sigma_{fg}\Sigma_{gg}^{-1/2}$:

$$\Sigma_{ff}^{-1/2}\Sigma_{fg}\Sigma_{gg}^{-1/2} = \tilde{U}D\tilde{B}^T \tag{2.5}$$

In this equation, Σ is the covariance matrix and D is a diagonal matrix containing the square of canonical correlation coefficients.

Singular Vector Canonical Correlation Analysis In order to reduce the effect of noise in measuring the similarity between representational spaces in CCA, Raghu et al. [2017] proposes Singular Vector Canonical Correlation Analysis (SVD), which applies SVD to select the most important directions in the representational spaces before applying CCA.

Projection-Weighted Canonical Correlation Analysis Having the canonical correlation coefficients, assuming all canonical variables are equally important, we can use their mean as an indicator of the similarity between two representational spaces. Morcos et al. [2018] argues that this way of measuring similarity could potentially underestimate the similarity between the high dimensional representations obtained from layers of neural networks. Morcos et al. [2018] proposed Projection-Weighted CCA that assigns weights to the canonical correlation coefficient based on the importance of their corresponding canonical variables to the underlying representational spaces. In [Morcos et al., 2018] the importance of the canonical variables is estimated based on the proportion of the original representations that they account for and they measure this in terms of dot product similarity of the canonical variables and the original representations.

2.1.2 Relational Similarity

To overcome the issue of comparing representations obtained from heterogeneous models and also to deal with trivial differences between representational spaces, we can represent the representational spaces in terms of the relations between different examples as they are embedded in them. I.e., having a set of examples, and their corresponding representations in the two spaces, instead of directly comparing the representations of each example, we first model the relation between the examples in each space and then measure their relational similarity, i.e., the similarity of similarities.

Similarity Matrix For a d dimensional representational space and a set of N examples, the similarity or co-variance matrix is a squared symmetrical matrix, $\Sigma_{N\times N}$, where Σ_{ij}, represents the similarity/co-variance between representations of examples i and j.

Representational Similarity Analysis Representational similarity analysis (RSA) is a technique which allows us to compare heterogeneous representational spaces [Laakso and Cottrell, 2000] and is a common technique in cognitive neuroscience because it allows researchers to study the relation between patterns of activation in the brain and representations of stimuli in a computational model [Kriegeskorte et al., 2008].

Given a set of N examples from a dataset, and their corresponding representation $F(X)$ and $G(X)$, RSA consists of two steps:

- Computing the similarity matrices Σ^f and Σ^g for F and G, where $\Sigma^f_{i,j}$ and $\Sigma^g_{i,j}$ are the similarities between the representation of x_i and x_j in each representational space respectively.
- Computing the correlation or similarity between the similarity matrices.

Often, in the first step, cosine distance is used to measure the similarity between the representations within each space and in the second step a correlation metric such as Pearson-R is employed. However, generally, at each stage, one can use any similarity metric that is more appropriate based on the nature of the representations.

Centered Kernel Alignment Kornblith et al. [2019] shows that if in standard RSA, the inner product is used as the similarity measure as shown in equation 2.6, the similarity between representational similarity matrices reduces to Hilbert-Schmidt Independence Criterion (HSIC).

$$\text{Sim}(F, G) = tr(FF^T GG^T) \tag{2.6}$$

HSIC is a pairwise similarity metric that measures the dependence of two variables. It generalizes 2.6 to inner products from reproducing kernel Hilbert spaces.

In practice and with finite samples n, and given K and L as kernel functions for F and G respectively, HSIC can be estimated by:

$$HSIC(K, L) = \frac{1}{(n-1)^2} tr(KHLH) \tag{2.7}$$

where $H, K, L \in R^{nxn}$, $K_{ij} := k(f_i, f_j)$, $L_{i,j} := l(g_i, g_j)$ and $H_{i,j} := \sigma_{ij} - \frac{1}{n}$.

For linear kernels, HSIC is equal to:

$$\frac{1}{(n-1)^2} tr(FF^T GG^T) = \left\| Cov(F^T, G^T) \right\|_F^2 \tag{2.8}$$

HSIC is not invariant to isotropic scaling, hence, Kornblith et al. [2019] proposes to use Centered Kernel Alignment (CKA), a normalized version of HSIC, to measure the

similarity between two representational spaces:

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K), HSIC(L, L)}} \tag{2.9}$$

One can use different kernels for computing CKA, Kornblith et al. [2019] tries linear and RBF kernels and they find no significant difference between these two kernels in their experiments. Since, this method is more widely used in machine learning literature, in our experiments that are conducted later, we report linear CKA.

2.2 Representational Stability Analysis

We introduce the notion of *Representational Stability Analysis* (ReStA), where we compare representational instances of the *same* model, while systematically varying a single model parameter. ReStA is measured as $\text{RSA}(L_{k|c_i}, L_{k'|c_j})$, where k and k' are layer/component ids and c_i and c_j are different conditions. This gives us a probe to measure the sensitivity of the representations to different factors.

This chapter presents the results of applying ReStA to different language encoding models such as GoogleLM, ELMO and variants of Transformer based language models. We investigate the relations between different components of the language encoding models and the type of information that is captured in the learned representations without making any explicit assumptions.

Varying Depth From prior work, we expect a relation between the depth of the layers and the level of abstraction of their representations. We study this intuition here empirically by analyzing the different layers of the models. Moreover, we investigate the impact of increasing the number of layers on the representational space of different variants of Transformer language models.

Varying Context Length Using language models to learn contextualized representations has been a significant milestone, enabling the application of representations obtained from pre-trained language models on a variety of downstream tasks in natural language processing. However, simply the fact that language models have access to the contextual information during training and inference, doesn't mean they will actually exploit this information. Hence, we use ReStA to understand the role of context and study how and where the models integrate information over time. To do so, we modify the amount of context provided to the models to obtain the contextualized word representations. We do this at the sentence level. Thus, for the context length of 0, we only feed the target words to the models; For context length 1, we feed all the previous words in the current sentence to the models. For context length i where $i > 1$, in addition to

Table 2.1: Details of the third-party computational models used in the experiments, including a brief characterization of the optimization objective, the training corpus, and the dimensionality of representations we extract from them. Except for Google-LM and ELMO, all the other models are the pre-trained checkpoints available in the hugging-face transformers library.

Model	Objective	Corpus	Rep.Dim.	Num.Layers	Architecture
albert-base-v2 [Lan et al., 2019]	MLM Sentence Ordering Prediction	Book Corpus English Wikipedia	768	12	Transformer encoder with weight sharing in depth
albert-large-v2 [Lan et al., 2019]	MLM Sentence Ordering Prediction	Book Corpus English Wikipedia	1024	24	Transformer encoder with weight sharing in depth
longformer-base [Beltagy et al., 2020]	MLM on long documents	Books Corpus English Wikipedia Subset of the Realnews dataset with documents longer than 1,200 tokens One third of the Stories Corpus	768	12	Transformer encoder with a combination of local and global attention
longformer-large [Beltagy et al., 2020]	MLM on long documents	Books Corpus English Wikipedia Subset of the Realnews dataset with documents longer than 1,200 tokens One third of the Stories Corpus	1024	24	Transformer encoder with a combination of local and global attention
bert-base [Devlin et al., 2019]	MLM Next Setence Prediction	Book Corpus English Wikipedia	768	12	Transformer encoder
bert-large [Devlin et al., 2019]	MLM Next Sentence Prediction	Book Corpus English Wikipedia	1024	24	Transformer encoder
electra-base-discriminator [Clark et al., 2020]	Replaced token detection	Book Corpus English Wikipedia	768	12	Transformer encoder
electra-large-discriminator [Clark et al., 2020]	Replaced token detection	Book Corpus English Wikipedia ClueWeb Gigaword	1024	24	Transformer encoder
roberta-base [Liu et al., 2019b]	MLM with dynamic masking	Book Corpus English Wikipedia	768	12	Transformer encoder
roberta-large [Liu et al., 2019b]	MLM with dynamic masking	Book Corpus English Wikipedia	1024	24	Transformer encoder
openai-gpt [Radford et al., 2018]	CLM	Book Corpus	768	12	Transformer Decoder
gpt2-small [Radford et al., 2019]	CLM	WebText	768	12	Transformer Decoder
gpt2-medium [Radford et al., 2019]	CLM	WebText	1024	24	Transformer Decoder
gpt2-large [Radford et al., 2019]	CLM	WebText	1280	36	Transformer Decoder
Google-LM [Jozefowicz et al., 2016]	CLM	One Billion Word Benchmark	1024	1	LSTM with larger memory state
ELMO [Peters et al., 2018a]	CLM (bidirectional)	One Billion Word Benchmark	4096	2	Bi-LSTM

the current sentence we feed all the words in the last i sentences. We operate on the sentence level to feed the model with independently meaningful pieces of text.

Different Objectives and Different Network Architectures In our studies, we consider language models with different architectures (i.e., LSTMs and different variants of Transformers), and different language modelling objectives (e.g., next word prediction, masked language modelling and replaced token detection). By comparing the representational spaces of these models, under different conditions, we aim to understand if the combination of network architecture and objective have a significant role in the solutions these language models converge to.

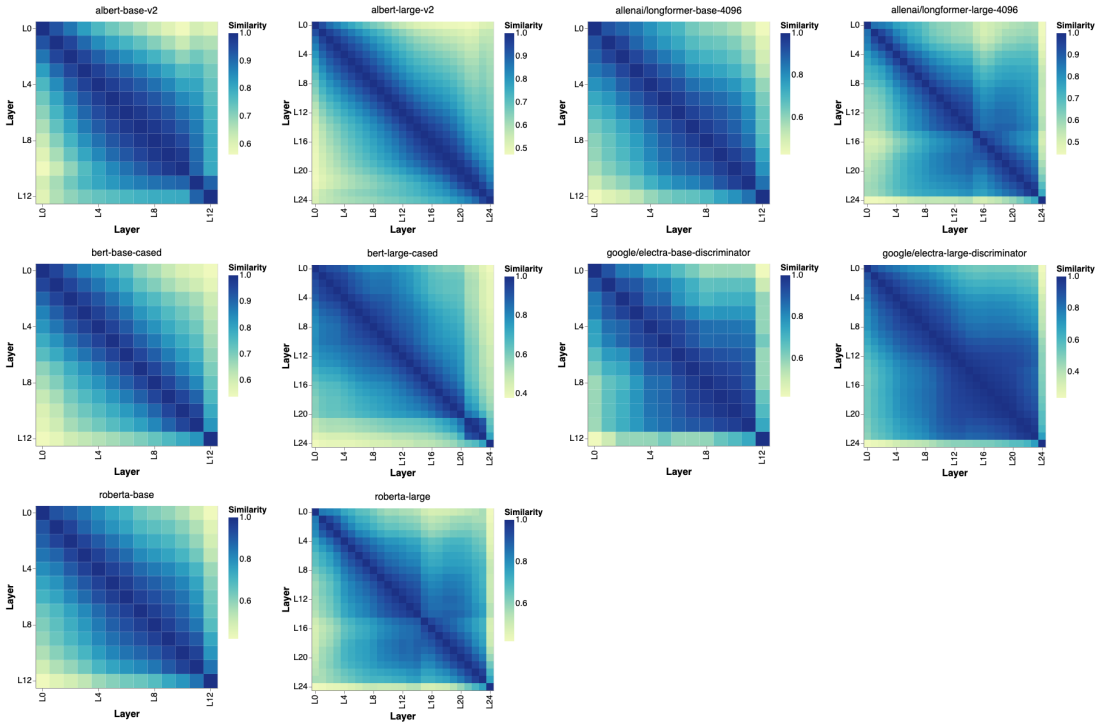


Figure 2.1: Representational similarity between layers of different Bert style transformer language models. The similarity score is measured in terms of CKA, Equation 2.9, over 100 sentences (words) from the Penn Tree Bank dataset. On the diagonal, we see the similarity score of 1.0 since it indicates the representational similarity of each layer with itself. We also observe high similarity scores in off-diagonal elements close to the diagonal. This indicates a smooth transformation of representations across layers. Additionally, we see that for all these models the last layer is the least similar to all the other layers.

2.2.1 Effect of Depth

In this section, we study the representations obtained from different layers of auto-regressive and non-auto-regressive language models and investigate how the representational spaces of various language models evolve across layers.

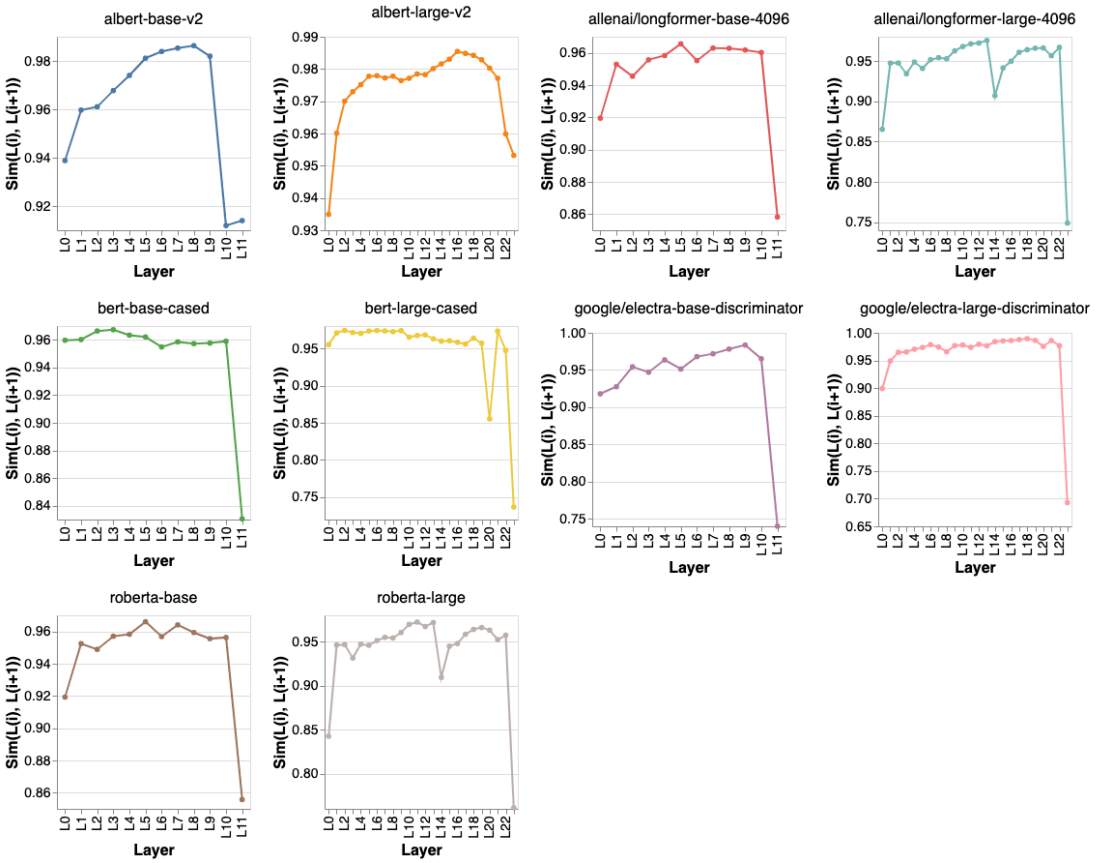


Figure 2.2: Representational similarity between subsequent layers of different Bert style transformer language models. The similarity score is measured in terms of CKA, Equation 2.9, over 100 sentences (words) from the Penn Tree Bank dataset.

Figure 2.1 illustrates the similarity between different layers for different variants of Transformer language models trained with the masked language modelling objective (BERT style) with different model sizes (base with 12 layers and large with 24 layers, in terms of CKA score. To show, more clearly how much the representations change in each layer, Figure 2.2, depicts the CKA similarity score between subsequent layers of Bert-style transformer models.

We observe that neighboring layers have more similar representations and the representational spaces gradually diverge as we move across the layers. Interestingly, between some of the layers, we observe a stronger divergence. This is more apparent for the last layer of all models except Alberta. This can be rooted in the fact that the last layer of the model carries most of the responsibility of capturing the task (objective) relevant features. For Alberta, the changes even at the last layer are less significant, which can be explained by the weight-sharing mechanism across layers. It appears that sharing weights in depth can enforce effective computations to distribute across the layers in a more uniform manner.

Additionally in some cases, e.g., Electra, Longformer, and Roberta, we can identify

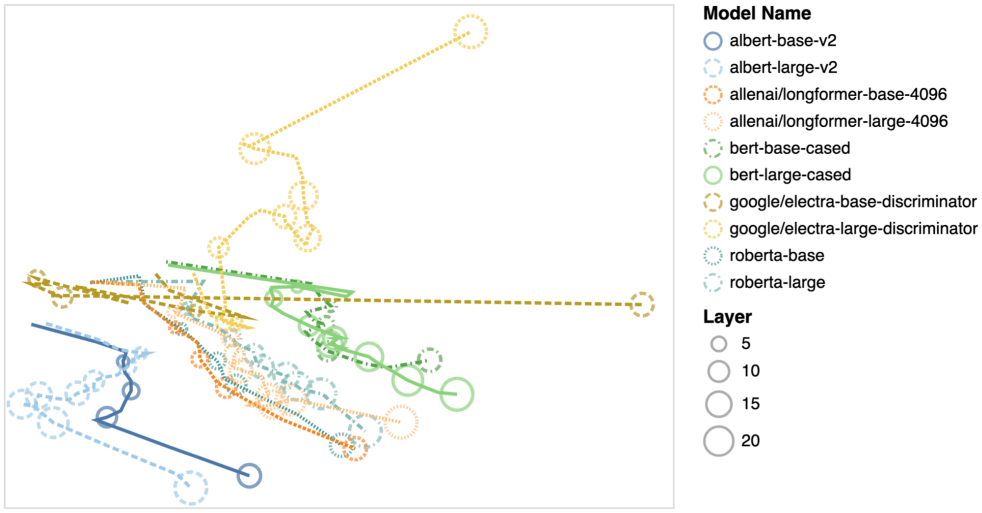


Figure 2.3: 2d projection of representational spaces of layers of Bert style transformer language models based on their pairwise CKA similarity. We observe that models from the same family (same architecture, same objective, same training data), with different sizes, follow similar trajectories, where shorter models take larger steps across the layers.

two or more blocks of layers, where layers within a block are more similar. The block boundaries are clearer in larger models (models with more layers), where generally the evolution across the layers is smoother. It is not obvious from this analysis what these blocks correspond to, but it is a curious case for further investigations. For example, does this mean that we can prune these layers without losing any performance?

Figure 2.3 is a 2d projection of all layers of all the Bert-style language models we are investigating based on their pairwise CKA similarity score. We can see that (a) layers of models with the same architecture and training objective but a different number of layers follow the same path for most of the cases, (b) the representations obtained from a different layer of different models diverge as we get closer to the penultimate layer which could be an indicator that the representations become more task-specific as they evolve across the layers, and (c) Representational space of Electra, which has a slightly different objective are further away and are following a completely different trajectory compared to the rest of the models. We also, observe that the representational spaces of Roberta and Longformer models are relatively tied together. This can simply be a side effect of the fact that Longformer models are initialized with Roberta checkpoints, and it is interesting to note that the representational spaces are relatively stable throughout the training process of Long-former models. They remain similar to Roberta, even though the ability of these models to deal with longer-range sequences is improved significantly.

Based on these figures, it seems, the effect of increasing the number of layers is merely a smoother transition across layers, and when we track how representations change

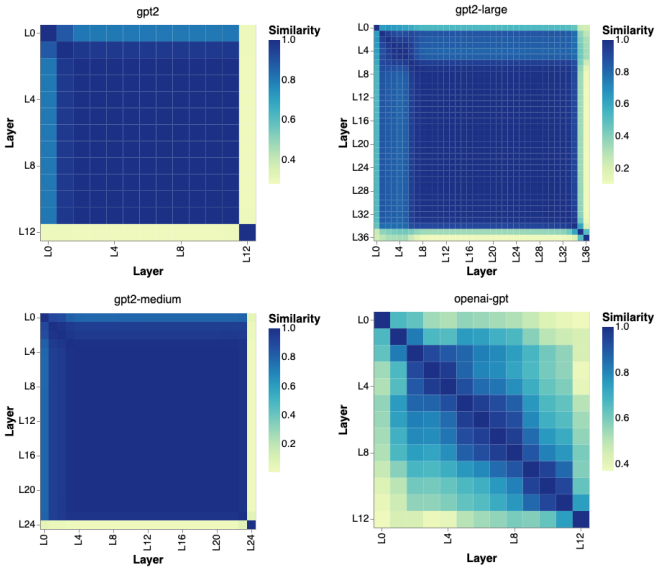


Figure 2.4: Representational similarity between all layers of different auto-regressive transformer language models. The similarity score is measured in terms of CKA, Equation 2.9, over 100 sentences (words) from the Penn Tree Bank dataset.

across the layers, representations obtained from different layers of the models with the same architecture, same training objective, and same training data that differ only in hyper-parameters such as depth and width converge not only through the same path but also to the same final point.

For auto-regressive transformer language models, in Figures 2.4 and 2.5, we see that for GPT-2 the representational spaces are less sensitive to the depth of the layer for the most part. I.e., there is a significant change in the first and the last layer and the middle layers are structured into one or a few blocks. On the other hand, OpenAi-GPT seems to have slightly different behaviour, i.e. the similarity scores between the intermediate representations are relatively lower. It is intriguing how scaling the size and diversity of datasets from GPT to GPT-2 results in a solution with completely different characteristics. One possible explanation here could be that as the diversity of the data for GPT-2 is much higher, its representational space has to be much denser which could generally result in the distance between representations being smaller. If this is the case, then the observation here is mostly revealing a bias of the similarity metric we have applied, and a potential solution to this issue could be normalizing the distances to take the density of the representational space into account.

2.2.2 Sensitivity to Context Length

Different language models, depending on their inductive biases and the data that they are trained on, could potentially implement different strategies to incorporate context. We can characterise the contextualization process by the general context sensitivity of the

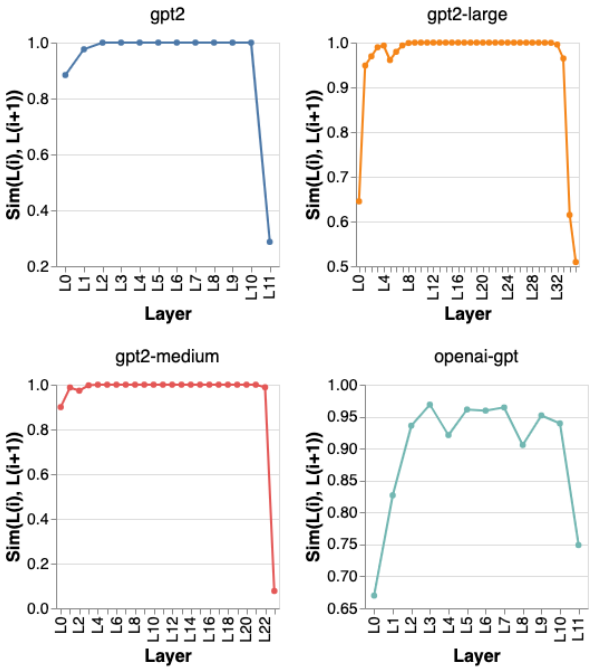


Figure 2.5: Representational similarity between subsequent layers of different auto-regressive transformer language models. The similarity score is measured in terms of CKA, Equation 2.9, over 100 sentences (words) from the Penn Tree Bank dataset.

models as well as the difference between the context-sensitivity of different components of these models.

Figure 2.6, illustrates a few different possible ways that a language model with multiple layers can mix contextual information with the information from each specific token. For example, every layer can contribute equally in incorporating the context into the representations, Figure 2.6a, the contextualization can happen in the final layers of the model after the individual representations are processed for a number of steps, Figure 2.6b, or initial layers of the model can contribute the most in the contextualization process, Figure 2.6c. These are just a few different examples and in practice, models could implement more complex contextualization mechanisms depending on their architecture, objective function, training data and other factors.

We aim to use the representational stability analysis technique for investigating the contextualization process in language models. We focus on the length of the prior context presented to the model as the condition. Varying the amount of context allows us to quantify the degree of context-dependence of different neural language models, and the different components of those models. If internal representations are similarly organized regardless of how much additional context is presented to the model, context-dependence is low. If, on the other hand, representations change with each additional amount of context included, context-dependence is high. Using this approach, we find intriguing differences between different neural language models (GoogleLM, ELMO,

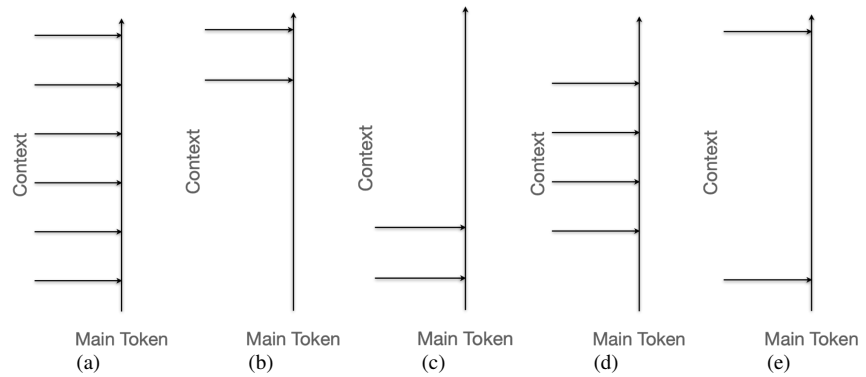


Figure 2.6: Symbolic diagram of different strategies for incorporating context in language models.

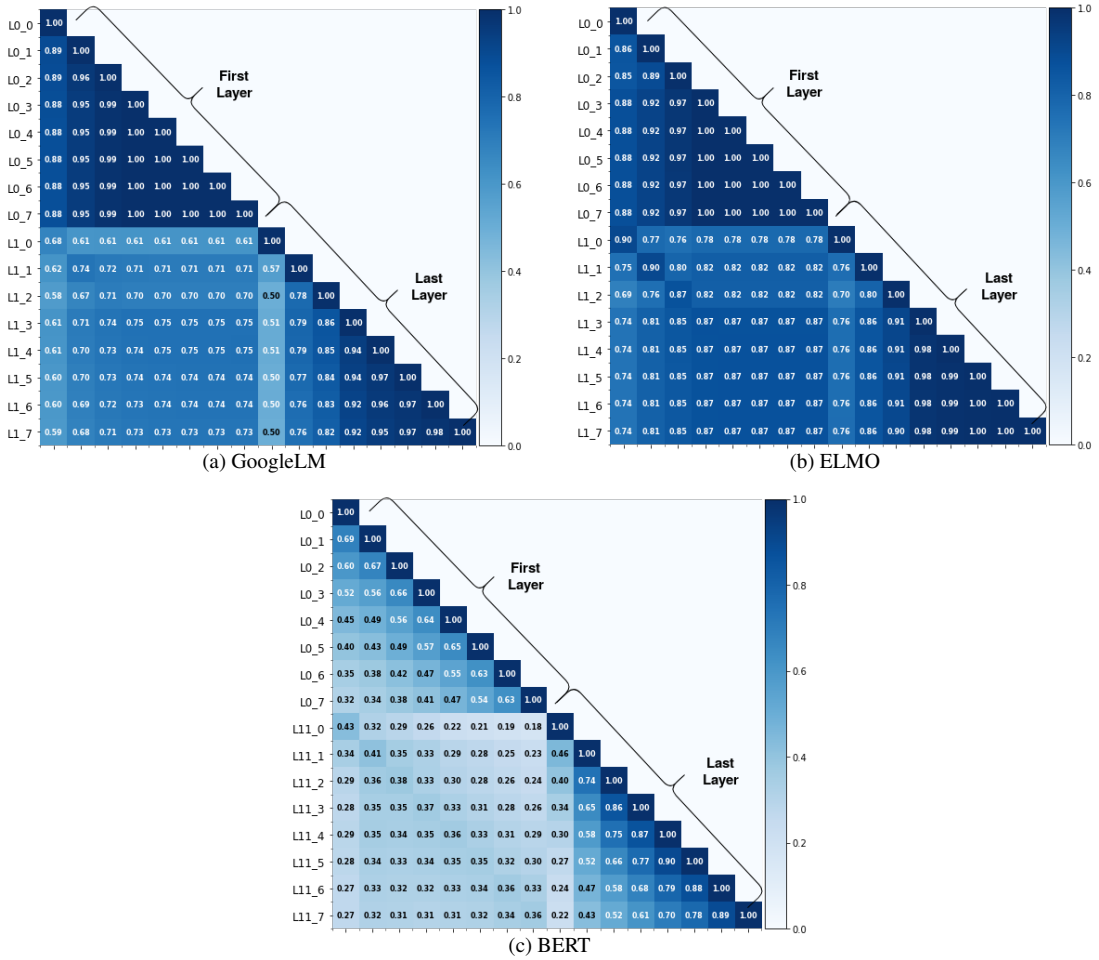
BERT and the Universal Sentence Encoder; Table 2.1), and between the first and deeper layers of those models.

In Figures 2.7 and 2.9a, we see that for both LSTM-based models, GoogleLM and ELMO, the first layer, $L0$, is less sensitive to the changes in the context length compared to the last layer, $L1$, i.e. the representations are not affected anymore by increasing the context length to more than 3 sentences. A hierarchical encoding mechanism, where the first layer is responsible for encoding the local context and the second(last) layer is encoding more global information, can justify these results.

As we can see in Figure 2.7 and more clearly in Figure 2.8, for the LSTM-based models, we observe a higher degree of similarity between the two layers (~ 0.75 and ~ 0.80) compared to BERT (~ 0.35). This can be partly explained by the higher number of layers in BERT, i.e the first and the last layer are further apart. Moreover, the relation between the first and last layers is almost the same for all context lengths and for all these three models the two layers are most similar when provided with the same amount of context.

We can see in Figure 2.9a, that the sensitivity to the context length is more significant in the Transformer based models compared to LSTM-based models. In these models, the difference in the representations at different context lengths does not fade away as the context length increases but the rate of the changes becomes constant. As illustrated in Figures 2.9a and 2.7c we observe that in BERT, regardless of the current context length, adding more context leads to different representations. Since in self-attention layers, there is a direct connection between the representations at different positions, the higher degree of sensitivity to context length is not surprising. This is evidence that, for computing the representations of each position in the input, the representations from all positions, no matter how far they are, are in fact taken into account.

We also observe that, in BERT, the representations from the first layer, $L0$ are more context-dependent than those from the last layer, $L11$, for long context lengths (context lengths longer than 1 sentence). To further investigate this, we look into the context sensitivity of different layers of Transformer based language models, Figures 2.10



(a) GoogleLM (b) ELMO

(c) BERT

Figure 2.7: Representational similarity score between different layers of each model given different context lengths in terms of the number of previous sentences over the story words. In these plots, for example, `L1_c3` means representation from layer 1 when the context length is 3 sentences including the current sentence. When $c = 0$, the model only sees the current words and when $c = 1$ the model sees the current sentence up to the target word. Here darker means more similar. The values are averaged over the four story blocks and the standard deviation of all the values across the four blocks is below 0.002.

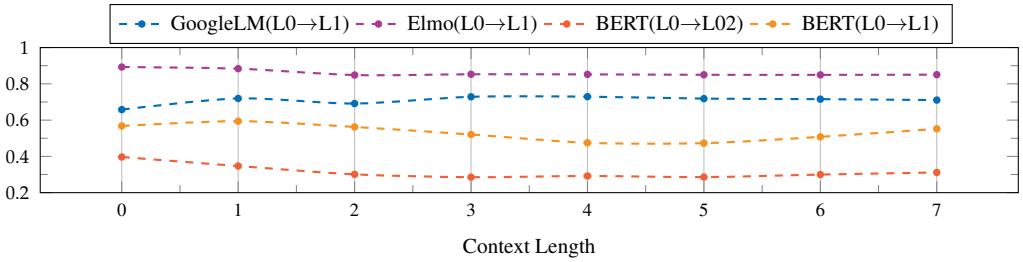
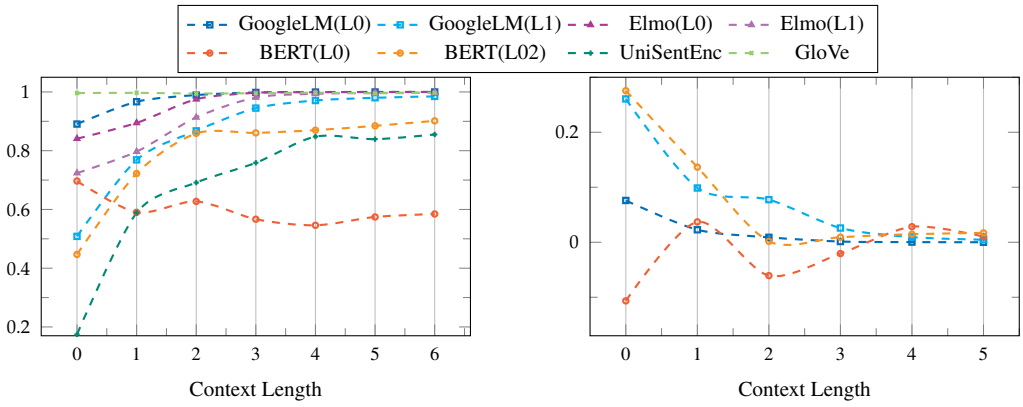


Figure 2.8: Layer similarities ($RSA(L_k\_c_i, L_{k+1}\_c_i)$. Here we show how increasing context length affects the similarity between different layers of the models.)



(a) Context Sensitivity ($RSA(L_{k\_}c_i, L_{k\_}c_{i+1})$)

(b) Changes in Context Sensitivity ($\delta RSA(L_{k\_}c_i, L_{k\_}c_{i+1})$)

Figure 2.9: Changes in RSA by increasing context length. (a) Shows how the amount of difference in the representational spaces changes by increasing the context length. (b) Shows for all models that we study, regardless of whether and how much their representations change by increasing context length, the amount of difference becomes almost constant after context length of 3 sentences. Note that in (b), we have scaled the plot and removed some of the models to increase the readability.

to 2.15. Observing the patterns in these plots we can hypothesize about the potential contextualization process implemented by these models.

We can see, in Figures 2.10 and 2.11, for both OpenAI-GPT and GPT-2 (small), all layers are more or less sensitive to changes in context length when the context is short (when we increase context length from 0 to 1 sentence). The main difference between GPT-2 (small) and OpenAI-GPT is the scale and diversity of the dataset they are trained with. GPT-2 is trained on a much larger and diverse dataset. Evidently, this has led to higher degrees of context-sensitivty for shorter context lengths in GPT-2 in all layers, and more strongly in the first and last layers. Additionally, OpenAI-GPT, the model that is trained with smaller scale data, shows higher degrees of context sensitivity in the intermediate layers when the context is longer. Without further investigations, it is not possible to explain these observations beyond what they are: scaling dataset, impacts the solutions these models converge to as characterized by how they incorporate context. Since, empirically, it is shown that GPT-2 performs better and has a better generalization power than OpenAI-GPT, one can argue that sensitivity to longer context in OpenAI-GPT could be a sign of memorization or as mentioned earlier this observation could be biased since the method is not sensitive to the density of points in the representational space.

In Figure 2.12, we see the context sensitivity pattern of different layers of BERT. Note that we see a similar pattern to Figure 2.7c. While the first layer is more sensitive to changes in context length, when context length is already longer than a sentence, the representations in the last layer change more dramatically when the context length increases from 0 to 1 sentence. Different context sensitivity patterns in the initial and final layers of BERT could be an indicator of the different roles that context plays in

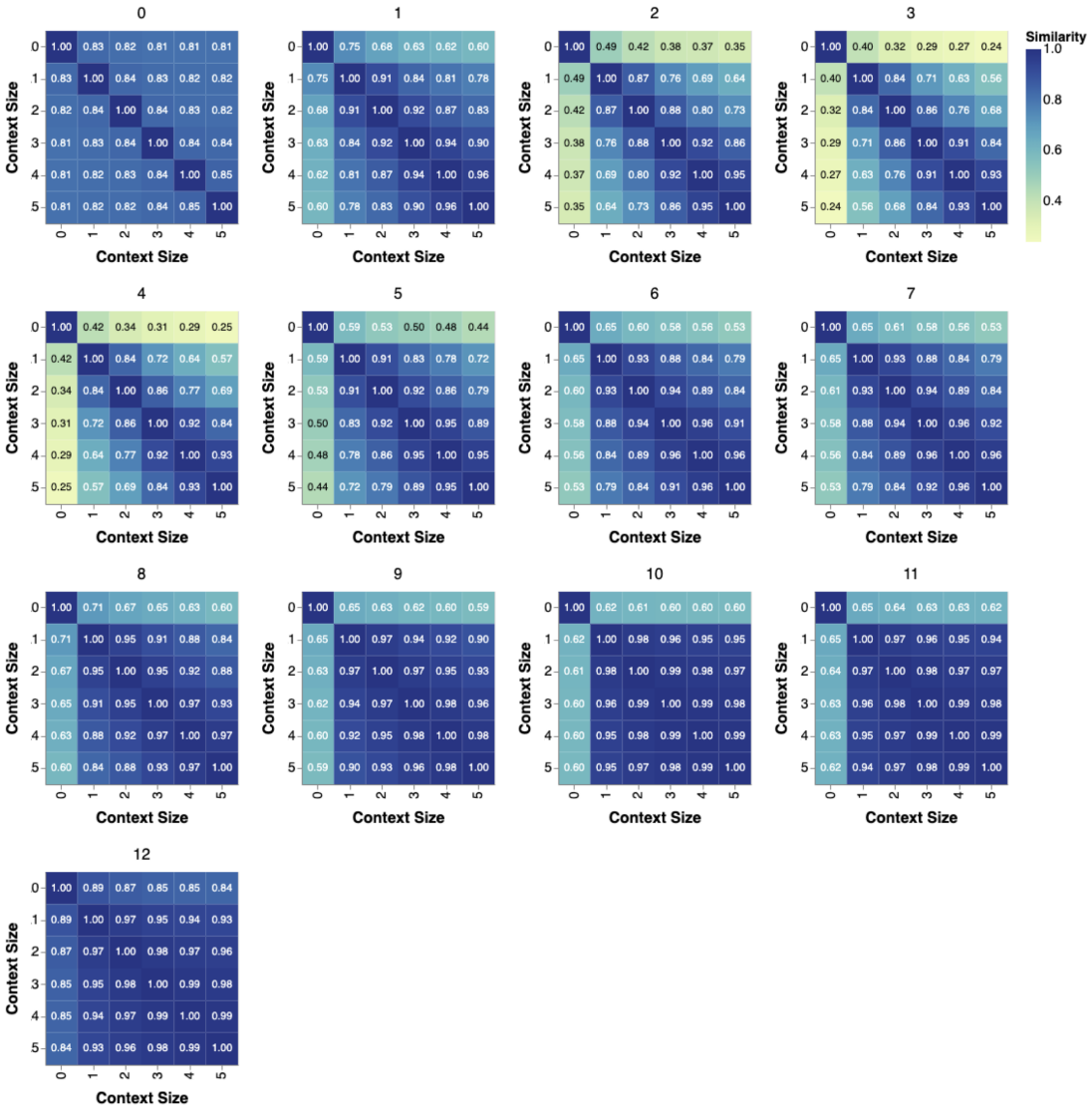


Figure 2.10: Context sensitivity of different layers of OpenAI GPT evaluated on 100 sentences from 100 different paragraphs from Book Corpus. We observe that lower intermediate layers are most sensitive to changes in context length.

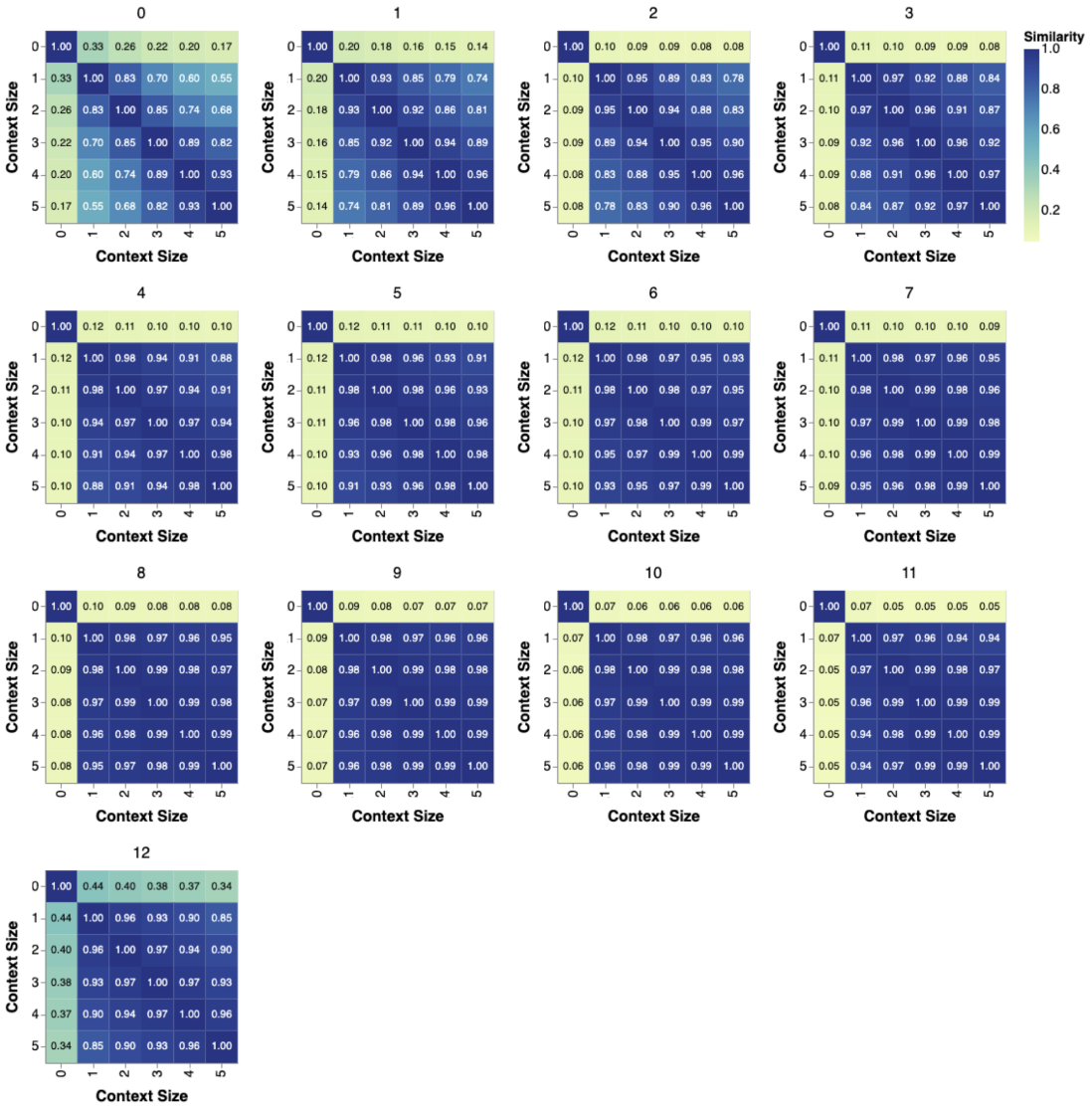


Figure 2.11: Context sensitivity of different layers of GPT2 evaluated on 100 sentences from 100 different paragraphs from Book Corpus. We observe that the first layer is most sensitive to the changes in context length when context length is longer (more than 1 sentence), and all the intermediate layers are equally sensitive to changes in context length for short context (change from none to 1 sentence.)

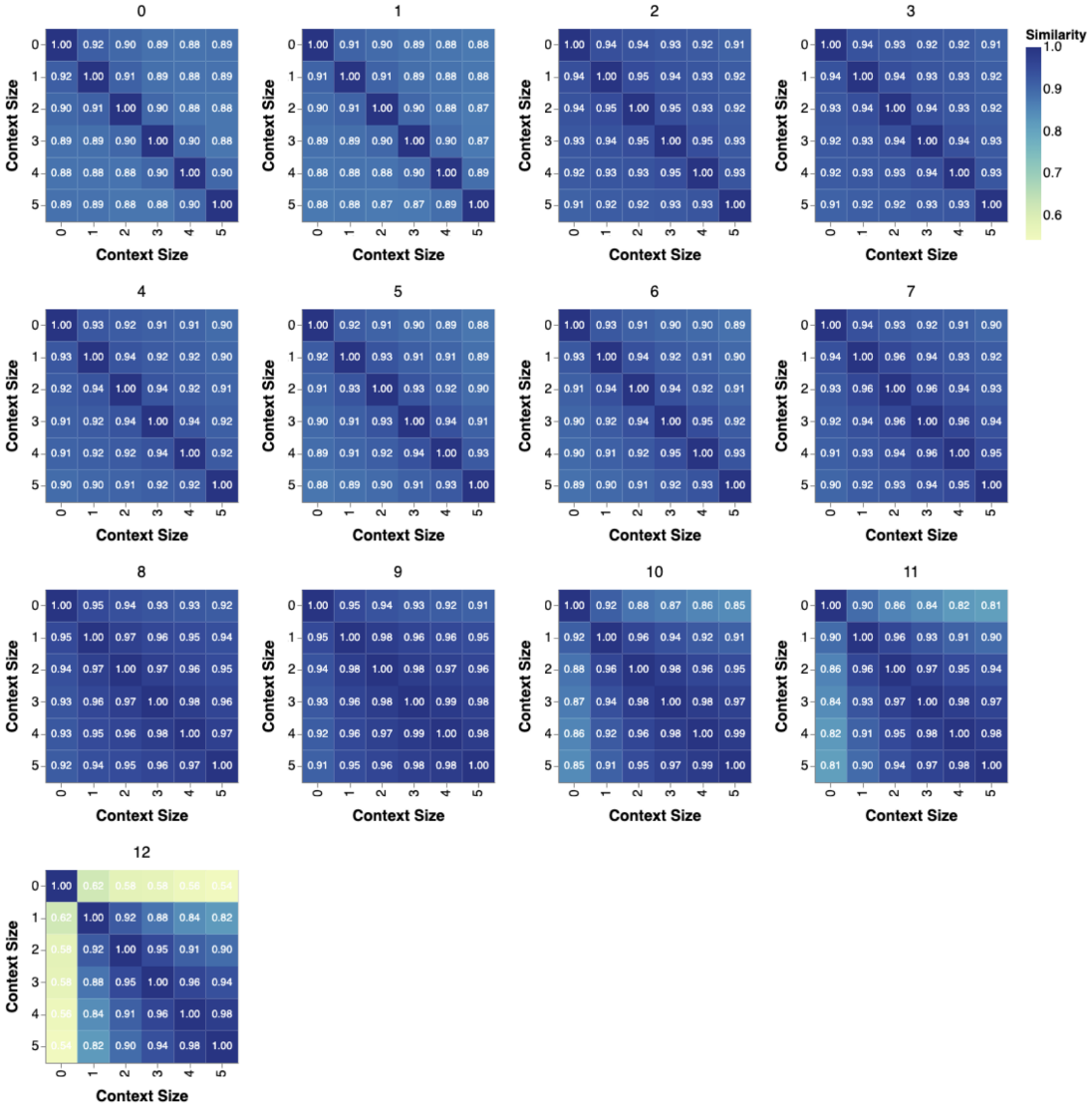


Figure 2.12: Context sensitivity of different layers of BERT evaluated on 100 sentences from 100 different paragraphs from Book Corpus.

these layers, e.g., in the initial layer the role of context is to help to disambiguate the meaning of each token, whereas in the final layers, and more specifically last layer the role of context depends on the objective which here is predicting the masked tokens.

Comparing BERT and Albert, Figures 2.12 and 2.13, investigating the role of architectural inductive biases, we see a significant difference. In Albert, the contextualization process is almost equally distributed across the layers (except for the last layer). This can be explained by the weight-sharing mechanism in Albert which biases this model to have similar computations across the layers. Additionally, in Albert, the context sensitivity increases through the first few layers and decreases again toward the last layer, Figure 2.13. This is interesting as we don't observe a similar trend in BERT or Long-former. We speculate, that the initial increase in context sensitivity, indicates a hierarchical encoding mechanism, similar to LSTM. This can be rooted in the partial recurrent inductive bias that this model has as a result of its parameter sharing in depth.

Comparing BERT and Electra, Figures 2.12 and 2.14, investigating the role of the objective function, we observe contrasting patterns. While in BERT the last layer is most sensitive to variation in context length for shorter context lengths, in Electra the last layer is the least context-sensitive layer (similar to Albert), and the initial and intermediate layers are the most context-sensitive.

Finally, in Long-former, Figure 2.15, the model that is designed and trained to be able to deal with long sequences, by incorporating specific attention patterns, and being trained on longer sequences, all layers are sensitive to both short and long context while the degree of context sensitivity gradually decreases as we move to toward the final layer.

We show here that using ReStA to characterize the way different models incorporate context into their representations we can reveal interesting differences in the solutions models with different inductive biases and training data converge to.

2.3 Representational Similarity Across Models

Finally, we study whether different language models have learned inherently different solutions by directly comparing their representational spaces. According to representational similarity scores, among the models that we study, shown in Figure 2.16, UniSentEnc seems to learn very different representations from ELMO, GoogleLM and BERT. While BERT and UniSentEnc are both Transformer based models, the representational space of BERT is more similar to the representations from ELMO and GoogleLM which are LSTM-based models. This can be due to the fact that ELMO, GoogleLM and BERT are trained with language modelling objectives, while UniSentEnc is trained on skip-thought and classification tasks and this could indicate the effect of the training objective on the representational spaces.

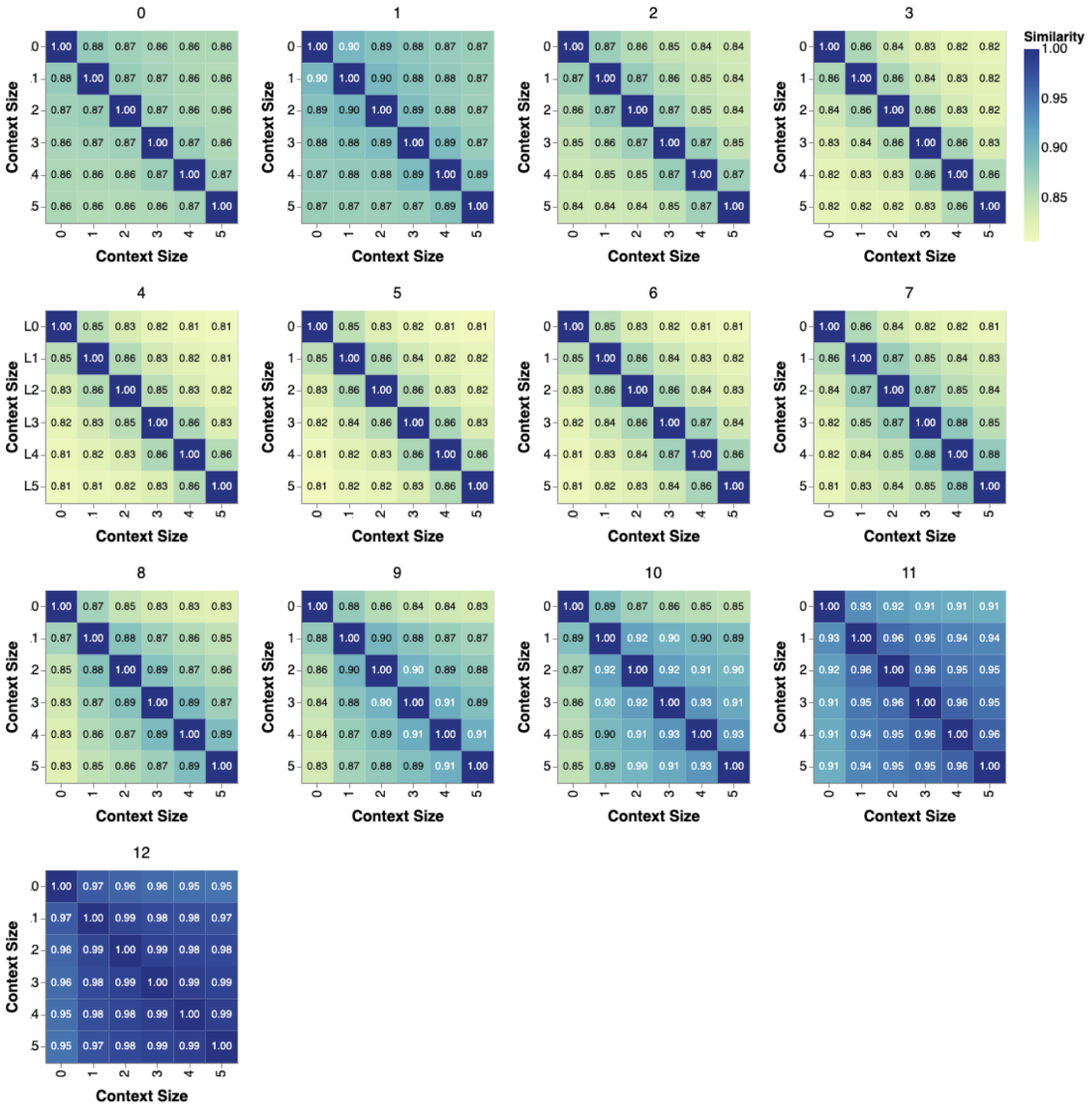


Figure 2.13: Context sensitivity of different layers of Albert evaluated on 100 sentences from 100 different paragraphs from Book Corpus. We observe that all layers except the last layer show significant context sensitivity, and the intermediate layers have the highest context sensitivity.

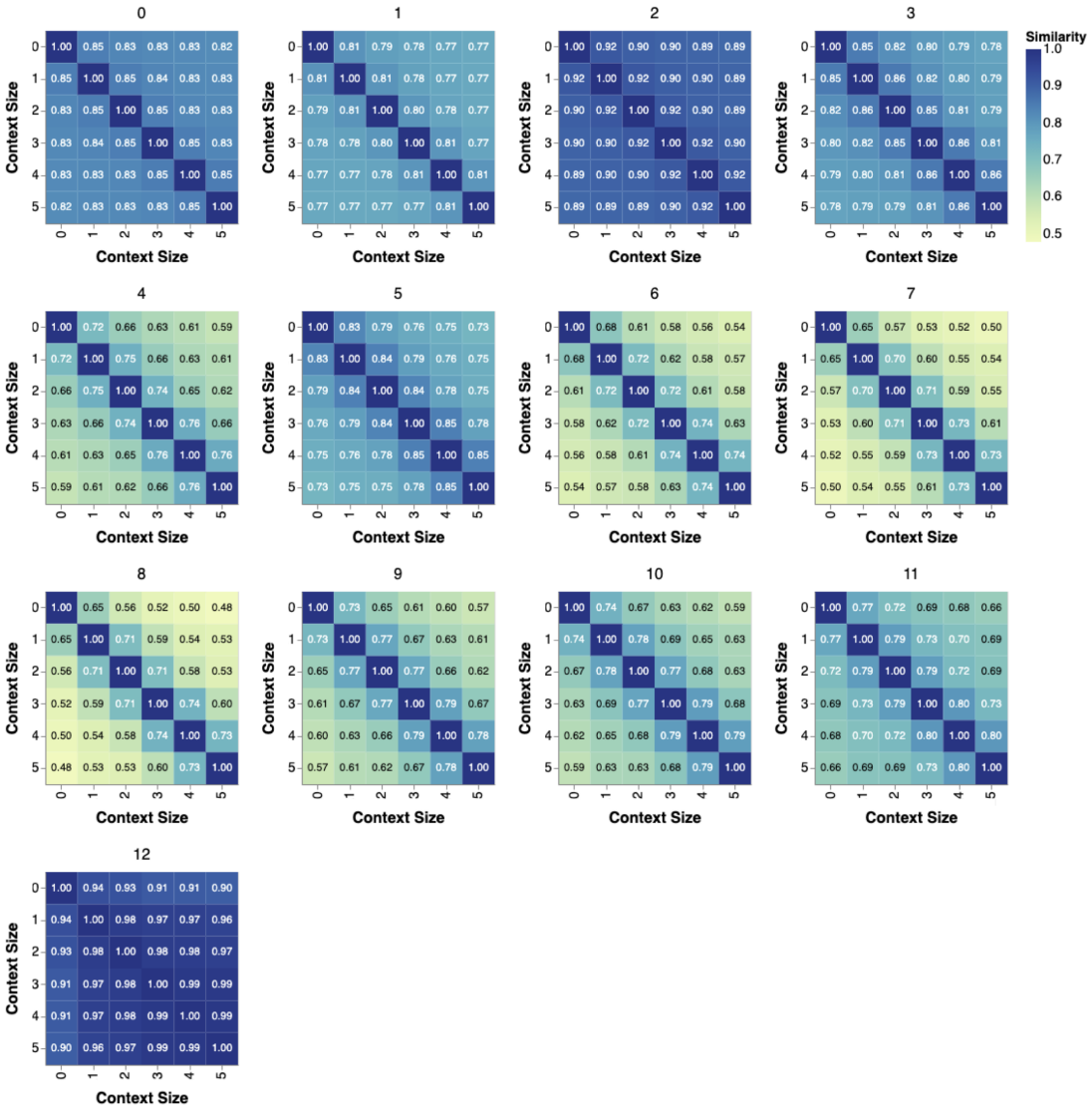


Figure 2.14: Context sensitivity of different layers of Electra evaluated on 100 sentences from 100 different paragraphs from Book Corpus. We observe that intermediate layers are most sensitive to changes in the context length.

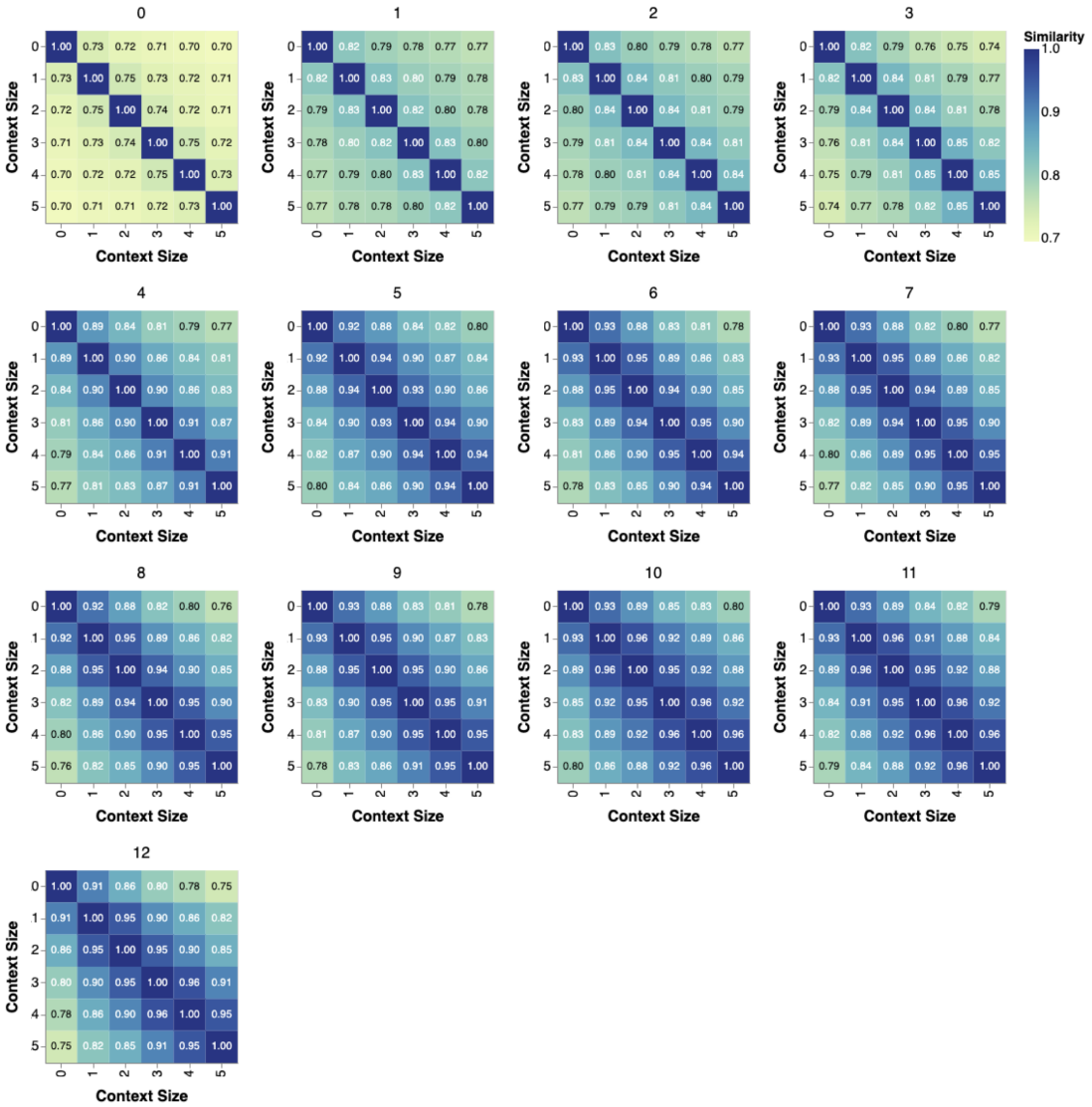


Figure 2.15: Context sensitivity of different layers of Long-former evaluated on 100 sentences from 100 different paragraphs from Book Corpus. We observe that all layers show relatively significant context sensitivity. The initial layer is the most context-sensitive layer and the context sensitivity gradually and slightly decreases as we move up across the layers.

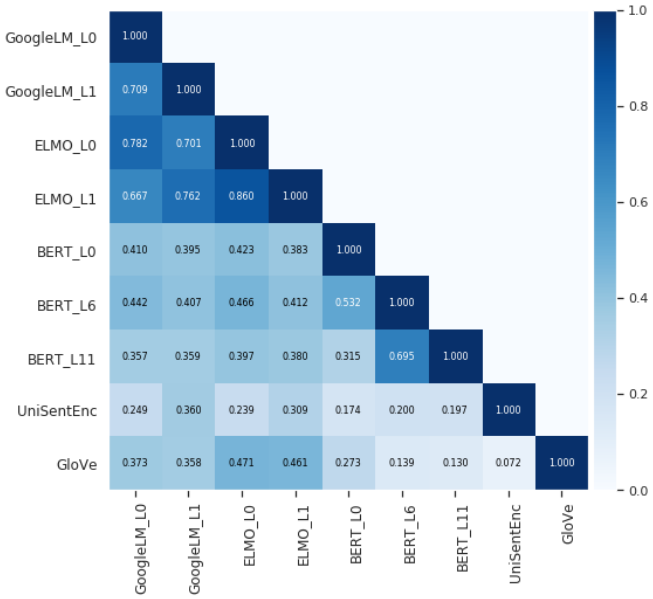


Figure 2.16: Representational similarity across models

2.4 Conclusion

In this chapter, we overview different techniques for comparing representational spaces of different models and introduce ReStA as a means to understand the impact of different inductive biases of different language models on the solution they converge to.

ReStA uses a representational similarity metric to measure the stability of the representations obtained from the models when a condition such as context length is changed. We argue that it is important to use a representational similarity metric that is easy to compute and that is invariant to trivial changes in the representational spaces. Hence, we use the standard representational similarity metric which compares the relational similarity of two spaces and is commonly used in neuroscience.

We explore this technique to characterise the solutions learned by different neural network-based language models. We find that both architectural differences, different training objectives and training data have a noticeable impact on the representations learned by the models and the way they change under different conditions. Not only, the representations obtained from the final layers of models with different architecture, language modelling objectives and training setup is different, but also they show different behaviours in terms of how their representations evolve across layers.

Generally, we observe that for most models, representations obtained from different layers of the models from the same family (same architecture, same objective, same training data) but different sizes (different depth and width) follow a similar trajectory and converge to very similar solutions in the last layers. The only difference is that shorter models take larger steps (the gap between representations of two consequent

layers is larger). On the other hand, factors such as training objective, scale and diversity of training data, and some architectural differences lead to significant differences in the characteristics of the solutions these models converge to. This is also evident in our analysis of applying ReStA to analyze different contextualization strategies implemented by different models, as well as our investigations of the evolution of representations across layers in different models.

One of our major observations, besides the general difference across different models, is that different forms of recurrence (in time or in depth), can lead the language models to incorporate a hierarchical approach of integrating contextual information. We dive deeper into the phenomena in chapter 6.

Our goal in this chapter is to show representational stability and similarity analysis can help us understand the characteristics of the solutions neural networks converge to. We believe this is a complementary technique to diagnostic classification and other probing techniques, to reveal the effect of different inductive biases of neural networks.

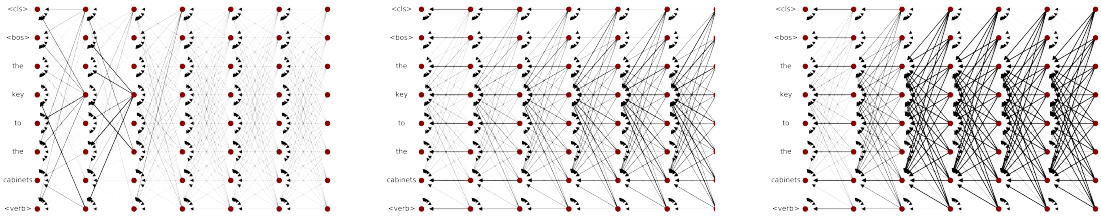
3

Investigating Attention Patterns

In the Transformer model, "self-attention" combines information from attended embeddings into the representation of the focal embedding in the next layer. Thus, across layers of the Transformer, information originating from different tokens gets increasingly mixed. This makes attention weights unreliable as explanations of the importance of input tokens across the layers. In this chapter, we consider the problem of quantifying this flow of information through self-attention. We propose two methods for approximating the attention to input tokens given attention weights, attention rollout and attention flow, as post hoc methods when we use attention weights as the relative relevance of the input tokens. We show that these methods give complementary views on the flow of information, and compared to raw attention, both yield higher correlations with importance scores of input tokens obtained using an ablation method and input gradients.

This chapter is based on the following paper.

- Samira Abnar and Willem Zuidema. 2020. Quantifying Attention Flow in Transformers. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4190–4197, Online. Association for Computational Linguistics.
- List of contributions is as follows. Samira Abnar: Designing and running the experiments, Writing the paper. Mostafa Dehghani: Designing the experiments, Reviewing and revising the paper. Willem Zuidema: Supervision and guiding the research, Reviewing and revising the paper.



(a) Embedding attentions (b) Attention rollout (c) Attention flow

Figure 3.1: Visualisation of attention weights.

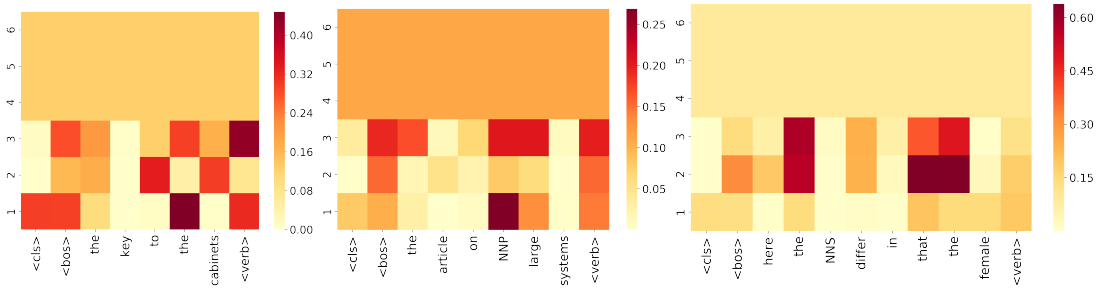


Figure 3.2: Raw Attention maps for the `CLS` token at different layers.

3.1 Quantifying Attention Flow in Transformers

Attention [Bahdanau et al., 2015, Vaswani et al., 2017] has become the key building block of neural sequence processing models, and visualizing attention weights is the easiest and most popular approach to interpret a model's decisions and to gain insights about its internals [Chen and Ji, 2019, Clark et al., 2019, Coenen et al., 2019, Dehghani et al., 2019, Lee et al., 2017, Rocktäschel et al., 2016, Vaswani et al., 2017, Wang et al., 2016, Xu et al., 2015]. Although it is wrong to equate attention with explanation [Jain and Wallace, 2019, Pruthi et al., 2019], it can offer plausible and meaningful interpretations [Vashishth et al., 2019, Vig, 2019, Wiegreffe and Pinter, 2019]. In this chapter, we focus on problems in interpreting attention weights arising when we move to the higher layers of a model, due to the lack of token identifiability of the embeddings in higher layers [Brunner et al., 2020].

We propose two simple but effective methods to compute attention scores to input tokens (i.e., *token attention*) at each layer, by taking raw attentions (i.e., *embedding attention*) of that layer as well as those from the precedent layers. These methods are based on modelling the information flow in the network with a *DAG* (Directed Acyclic Graph), in which the nodes are input tokens and hidden embeddings, edges are the attentions from the nodes in each layer to those in the previous layer, and the weights of the edges are the attention weights.

The first method, *attention rollout*, assumes that the identities of input tokens are linearly combined through the layers based on the attention weights. To adjust attention weights, it rolls out the weights to capture the propagation of information from input tokens to intermediate hidden embeddings.

The second method, *attention flow*, considers the attention graph as a flow network. Using a maximum flow algorithm, it computes maximum flow values, from hidden embeddings (sources) to input tokens (sinks).

In both methods, we take the residual connection in the network into account to better model the connections between input tokens and hidden embedding. We show that compared to raw attention, the token attentions from attention rollout and attention flow have higher correlations with the importance scores obtained from input gradients as well as *blank-out*, an input ablation based attribution method. Furthermore, we visualize the token attention weights and demonstrate that they are better approximations of how input tokens contribute to a predicted output, compared to raw attention.

It is noteworthy that the techniques we propose in this chapter, are not intended to make hidden embeddings more interpretable, or to provide better attention weights for better performance. Rather we aim at computing effective attention weights that take token identity problem into consideration and can serve as a better diagnostic tool for visualization and debugging.

3.1.1 Experimental Setup and Problem Statement

In our analysis, we focus on the verb number prediction task, i.e., predicting the singularity or plurality of a verb of a sentence, when the input is the sentence up to the verb position. We use the subject-verb agreement dataset [Linzen et al., 2016]. This task and dataset are convenient choices, as they offer a clear hypothesis about what part of the input is essential to get the right solution. For instance, given "*the key to the cabinets*" as the input, we know that attending to "key" helps the model predict singular as output while attending to "cabinets" (an *agreement attractor*, with the opposite number) is unhelpful.

We train a Transformer encoder, with GPT-2 Transformer blocks as described in [Radford et al., 2019, Wolf et al., 2019] (without masking). The model has 6 layers, and 8 heads, with a hidden/embedding size of 128. Similar to Bert [Devlin et al., 2019] we add a `CLS` token and use its embedding in the final layer as the input to the classifier. The accuracy of the model on the subject-verb agreement task is 0.96. To facilitate replication of our experiments we will make the implementations of the models we use and algorithms we introduce publicly available at *https://github.com/samiraabnar/attention_flow*.

We start by visualizing raw attention in Figure 3.1a (like Vig 2019). The example given here is correctly classified. Crucially, only in the first couple of layers, there are some distinctions in the attention patterns for different positions, while in higher layers the attention weights are rather uniform. Figure 3.2 (left) gives raw attention scores of the `CLS` token over input tokens (x-axis) at different layers (y-axis), which similarly lack an interpretable pattern. These observations reflect the fact that as we go deeper into the model, the embeddings are more contextualized and may all carry similar information.

This underscores the need to track down attention weights all the way back to the input layer and is in line with findings of Serrano and Smith [2019], who show that attention weights do not necessarily correspond to the relative importance of input tokens.

To quantify the usefulness or comprehensiveness of raw attention weights, and the two alternatives that we consider in the next section,

besides input gradients, we employ an input ablation method, *blank-out*, to estimate an importance score for each input token. Blank-out replaces each token in the input, one by one, with `UNK` and measures how much it affects the predicted probability of the correct class. We compute the *Spearman's rank correlation* coefficient between the attention weights of the `CLS` embedding in the final layer and the importance scores from blank-out. As shown in the first row of Table 3.1, the correlation between raw attention weights of the `CLS` token and blank-out scores is rather low, except for the first layer. As we can see in Table 3.2 this is also the case when we compute the correlations with input gradients.

Table 3.1: SpearmanR correlation of attention based importance with blank-out scores for 2000 samples from the test set for the verb number prediction model.

	L1	L2	L3	L4	L5	L6
Raw	0.69±0.27	0.10±0.43	-0.11±0.49	-0.09±0.52	0.20±0.45	0.29±0.39
Rollout	0.32±0.26	0.38±0.27	0.51±0.26	0.62±0.26	0.70±0.25	0.71±0.24
Flow	0.32±0.26	0.44±0.29	0.70±0.25	0.70±0.22	0.71±0.22	0.70±0.22

Table 3.2: SpearmanR correlation of attention based importance with input gradients for 2000 samples from the test set for the verb number prediction model.

	L1	L2	L3	L4	L5	L6
Raw	0.53±0.33	0.16±0.38	-0.06±0.42	0.00±0.47	0.24±0.40	0.46±0.35
Rollout	0.22±0.31	0.27±0.32	0.39±0.32	0.47±0.32	0.53±0.32	0.54±0.31
Flow	0.22±0.31	0.31±0.34	0.54±0.32	0.61±0.28	0.60±0.28	0.61±0.28

3.2 Attention Rollout and Attention Flow

Attention rollout and attention flow recursively compute the token attentions in each layer of a given model given the embedding attentions as input. They differ in the assumptions they make about how attention weights in lower layers affect the flow of information to the higher layers and whether to compute the token attentions relative to each other or independently.

To compute how information propagates from the input layer to the embeddings in higher layers, it is crucial to take the residual connections in the model into account

as well as the attention weights. In a Transformer block, both self-attention and feed-forward networks are wrapped by residual connections, i.e., the input to these modules is added to their output. When we only use attention weights to approximate the flow of information in Transformers, we ignore the residual connections. But these connections play a significant role in tying corresponding positions in different layers. Hence, to compute attention rollout and attention flow, we augment the attention graph with extra weights to represent residual connections. Given the attention module with residual connection, we compute values in layer $l+1$ as $V_{l+1} = V_l + W_{att}V_l$, where W_{att} is the attention matrix. Thus, we have $V_{l+1} = (W_{att} + I)V_l$. So, to account for residual connections, we add an identity matrix to the attention matrix and re-normalize the weights. This results in $A = 0.5W_{att} + 0.5I$, where A is the raw attention updated by residual connections.

Furthermore, analyzing individual heads requires accounting for the mixing of information between heads through a position-wise feed-forward network in the Transformer block. Using attention rollout and attention flow, it is also possible to analyze each head separately. We explain in more detail in Appendix 3.2.1. However, in our analysis in this chapter, for simplicity, we average the attention at each layer over all heads.

Attention rollout Attention rollout is an intuitive way of tracking down the information propagated from the input layer to the embeddings in the higher layers. Given a Transformer with L layers, we want to compute the attention from all positions in layer l_i to all positions in layer l_j, where $j < i$. In the attention graph, a path from node v at position k in l_i, to node u at position m in l_j, is a series of edges that connect these two nodes. If we look at the weight of each edge as the proportion of information transferred between two nodes, we can compute how much of the information at v is propagated to u through a particular path by multiplying the weights of all edges in that path. Since there may be more than one path between two nodes in the attention graph, to compute the total amount of information propagated from v to u, we sum over all possible paths between these two nodes. At the implementation level, to compute the attentions from l_i to l_j, we recursively multiply the attention weights matrices in all the layers below.

$$\tilde{A}(l_i) = \begin{cases} A(l_i)\tilde{A}(l_{i-1}) & \text{if } i > j \\ A(l_i) & \text{if } i = j \end{cases} \tag{3.1}$$

In this equation, $\tilde{A}$ is attention rollout, A is raw attention and the multiplication operation is matrix multiplication. With this formulation, to compute input attention we set $j = 0$.

Attention flow In graph theory, a flow network is a directed graph with a "capacity" associated with each edge. Formally, given $G = (V, E)$ is a graph, where V is the set of nodes, and E is the set of edges in G; $C = \{c_{uv} \in \mathbb{R} \mid \forall u, v \text{ where } e_{u,v} \in E \wedge u \neq v\}$

denotes the capacities of the edges and $s, t \in V$ are the source and target (sink) nodes respectively; *flow* is a mapping of edges to real numbers, $f : E \rightarrow \mathbb{R}$, that satisfies two conditions: (a) *capacity constraint*: for each edge, the flow value should not exceed its capacity, $|f_{uv} \leq c_{uv}|$; (b) *flow conservation*: for all nodes except s and t the input flow should be equal to the output flow –the sum of the flow of outgoing edges should be equal to the sum of the flow of incoming edges. Given a flow network, a maximum flow algorithm finds a flow which has the maximum possible value between s and t [Cormen et al., 2009].

Treating the attention graph as a flow network, where the capacities of the edges are attention weights, using any maximum flow algorithm, we can compute the maximum attention flow from any node in any of the layers to any of the input nodes. We can use this maximum-flow-value as an approximation of the attention to input nodes.

In attention flow, the weight of a single path is the minimum value of the weights of the edges in the path, instead of the product of the weights. Besides, we can not compute the attention for node s to node t by adding up the weights of all paths between these two nodes, since there might be an overlap between the paths and this might result in overflow in the overlapping edges.

It is noteworthy that both of the proposed methods can be computed in polynomial time. $O(d * n^2)$ for attention rollout and $O(d^2 * n^4)$ for attention flow, where d is the depth of the model, and n is the number of tokens.

3.2.1 Single Head Analysis

For analysing the attention weights, with a multi-head setup, we could either analyze attention heads separately, or we could average all heads and have a single attention graph. However, we should be careful that treating attention heads separately could potentially mean that we are assuming there is no mixing of information between heads, which is not true as we combine information of heads in the position-wise feed-forward network on top of self-attention in a transformer block.

It is possible to analyse the role of each head in isolation from all other heads using attention rollout and attention flow. To not make the assumption that there is no mixing of information between heads, for computing the "input attention", we will treat all the layers below the layer of interest as single-head layers, i.e., we sum the attentions of all heads in the layers below. For example, we can compute attention rollout for head k at layer i as $\tilde{A}(i,k) = A(i,k)\bar{A}_(i)$, where, $\bar{A}_(i)$ is attention rollout computed for layer i with the single head assumption.

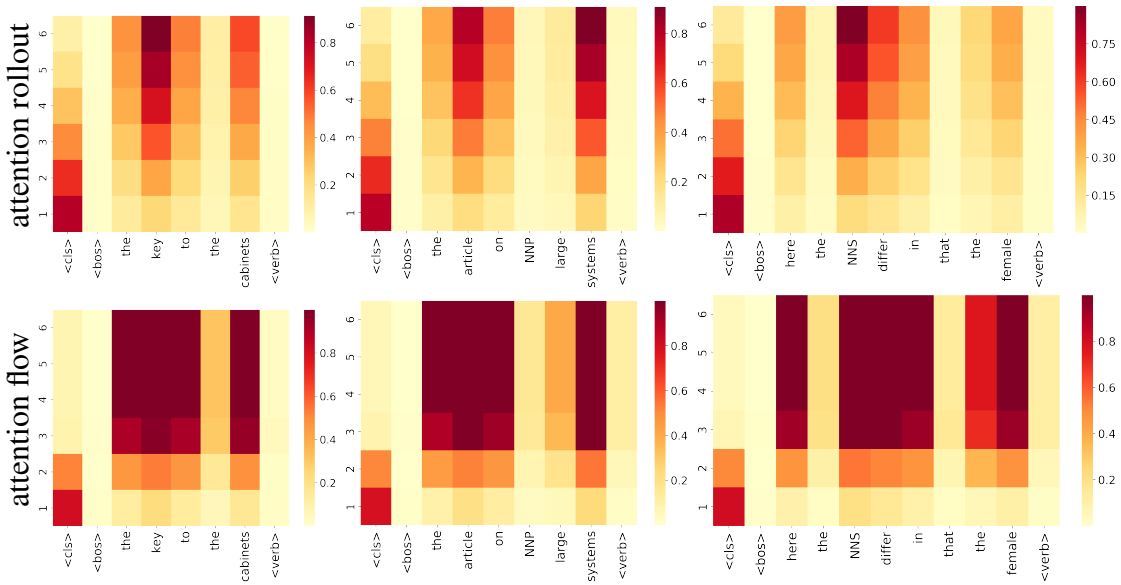


Figure 3.3: Attention maps for the CLS token

.

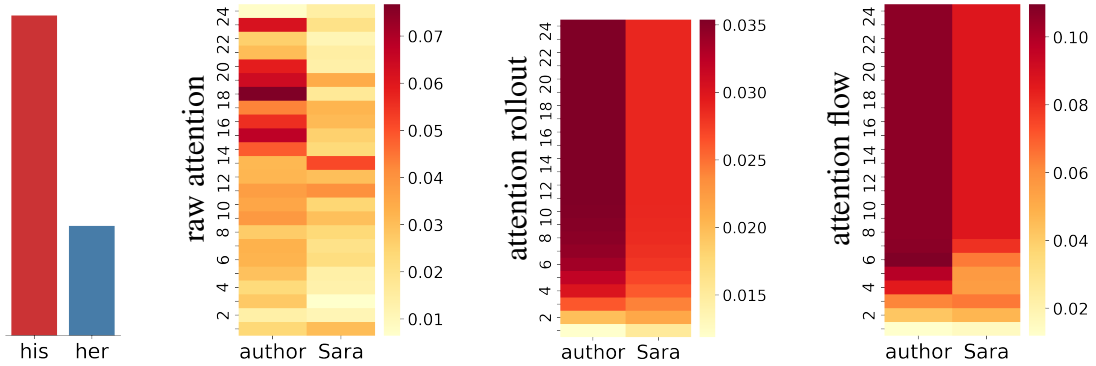
3.3 Analysis and Discussion

Now, we take a closer look at these three views of attention. Figure 3.1 depicts raw attention, attention rollout and attention flow for a correctly classified example across different layers. It is noteworthy that the first layer of attention rollout and attention flow are the same, and their only difference with raw attention is the addition of residual connections. As we move to the higher layers, we see that the residual connections fade away. Moreover, in contrast to raw attention, the patterns of attention rollout and attention flow become more distinctive in the higher layers.

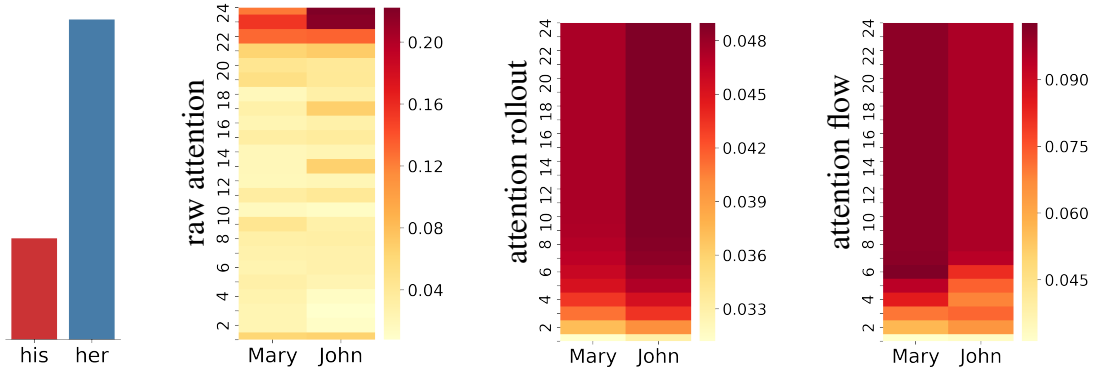
Figures 3.2 and 3.3 show the weights from raw attention, attention rollout and attention flow for the CLS embedding over input tokens (x-axis) in all 6 layers (y-axis) for three examples. The first example is the same as the one in Figure 3.1. The second example is "*the article on NNP large systems* <?>". The model correctly classifies this example and changing the subject of the missing verb from "article" to "article**s**" flips the decision of the model.

The third example is "*here the NNS differ in that the female* <?>", which is a miss-classified example and again changing "NNS" (plural noun) to "NNP" (singular proper noun) flips the decision of the model.

For all cases, the raw attention weights are almost uniform above layer three (discussed before). In the case of the correctly classified example, we observe that both attention rollout and attention flow assign relatively high weights to both the subject of the verb, "article' and the attractor, "systems". For the miss-classified example, both attention rollout and attention flow assign relatively high scores to the "NNS" token which is not the subject of the verb. This can explain the wrong prediction of the model.



(a) "*The author talked to Sara about* `mask` *book.*"



(b) "*Mary convinced John of* `mask` *love.*"

Figure 3.4: Bert attention maps. We look at the attention weights from the `mask` embedding to the two potential references for it, e.g. "author" and "Sara" in (a) and "Mary" and "John" in (b). The bars, at the left, show the relative predicted probability for the two possible pronouns, "his" and "her".

The main difference between attention rollout and attention flow is that attention flow weights are amortized among the set of most attended tokens, as expected. Attention flow can indicate a set of input tokens that are important for the final decision. Thus we do not get sharp distinctions among them. On the other hand, attention rollout weights are more focused compared to attention flow weights, which is sensible for the third example but not as much for the second one.

Table 3.3: SpearmanR correlation of attention based importance with input gradients for 100 samples from the test set for the DistillBERT model fine tuned on SST-2.

	Layer 1	Layer 3	Layer 5	Layer 6
Raw Attentions	0.12 ± 0.21	0.09 ± 0.21	0.08 ± 0.20	0.09 ± 0.21
Attention Rollout	0.11 ± 0.19	0.12 ± 0.21	0.13 ± 0.21	0.13 ± 0.20
Attention Flow	0.11 ± 0.19	0.11 ± 0.21	0.12 ± 0.22	0.14 ± 0.21

Furthermore, as shown in Table 3.1 and 3.2 both attention rollout and attention flow, are better correlated with blank-out scores and input gradients compared to raw attention, but attention flow weights are more reliable than attention rollout. The difference between these two methods is rooted in their different views of attention weights. Attention flow views them as capacities, and at every step of the algorithm, it uses as much of the

capacity as possible. Hence, attention flow computes the maximum possibility of token identities to propagate to the higher layers. Whereas attention rollout views them as proportion factors and at every step, it allows token identities to be propagated to higher layers exactly based on these proportion factors. This makes attention rollout stricter than attention flow, and so we see that attention rollout provides us with more focused attention patterns. However, since we are making many simplifying assumptions, the strictness of attention rollout does not lead to more accurate results, and the relaxation of attention flow seems to be a useful property.

At last, to illustrate the application of attention flow and attention rollout on different tasks and different models, we examine them on two pretrained BERT models. We use the models available at *https://github.com/huggingface/transformers*.

Table 3.3 shows the correlation of the importance score obtained from raw attention, attention rollout and attention flow from a DistillBERT [Sanh et al., 2019] model fine-tuned to solve "SST-2" [Socher et al., 2013], the sentiment analysis task from the glue benchmark [Wang et al., 2018]. Even though for this model, all three methods have a very low correlation with the input gradients, we can still see that attention rollout and attention flow are slightly better than raw attention. Note that, low correlation with blank-out (or any other empirical method to compute input attribution) is not necessarily a negative outcome, as blank-out itself is not a perfect method for input attribution.

Furthermore, in Figure 3.4, we show an example of applying these methods to a pre-trained Bert to see how it resolves the pronouns in a sentence. What we do here is to feed the model with a sentence, masking a pronoun. Next, we look at the prediction of the model for the masked word and compare the probabilities assigned to "her" and "his". Then we look at raw attention, attention rollout and attention flow weights of the embeddings for the masked pronoun at all the layers.

In the first example, in Figure 3.4a, attention rollout and attention flow are consistent with each other and the prediction of the model. Whereas, the final layer of raw attention does not seem to be consistent with the prediction of the models, and it varies a lot across different layers. In the second example, in Figure 3.4b, only attention flow weights are consistent with the prediction of the model.

3.4 Attention Span of Transformer Language Models

We can study attention patterns as a proxy to estimate the context sensitivity of transformer models, as a complementary technique to the representational stability analysis technique applied in chapter 2. We compute a metric we refer to as the "attention span score" of the models. The attention span score for each query token is computed as the weighted average of the relational positions of the key tokens (distance of the key tokens to the query token) where the weights are the attention scores obtained from the model. As showin in equation 3.2. In this equation, Q and K are the set of queries and

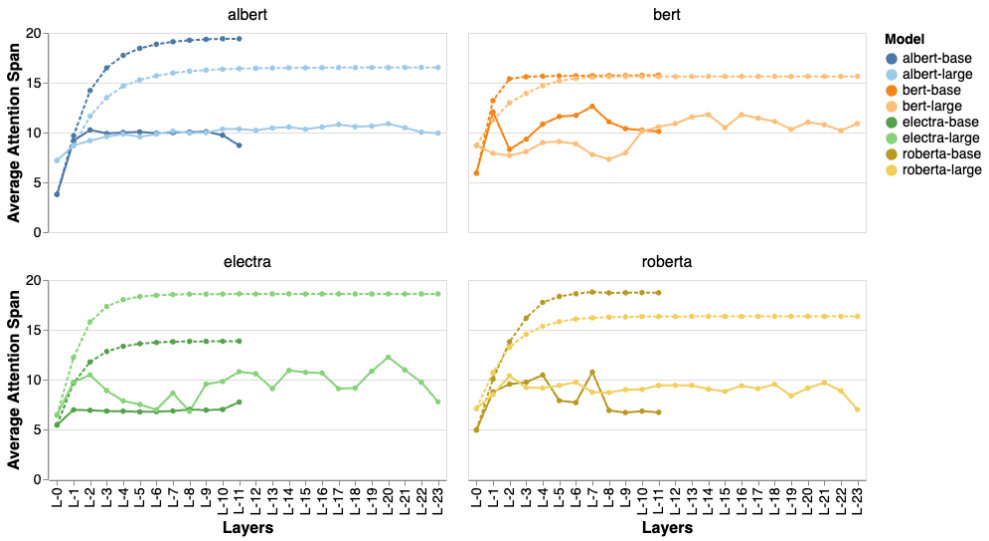


Figure 3.5: Attention span score based on raw attention and attention rollout for different layers of various Transformer language models trained on masked language modelling. In this picture, the attention span score is averaged for all heads in each layer. The attention span of a head is the average attention span of a given set of sentences. The attention span score for each sentence is the maximum of the attention span of its tokens, and the attention span score of each token is the weighted sum of its relative distances from the other tokens in the sentence where the weights are the attention scores. We observe that the attention span is increasing as we move toward the higher layers. That means layers closer to the output, potentially, take a wider context into account.

keys in the attention module, d_{qk} is the distance between the query node q and key node k, and A_{qk} is the attention score from q to k.

$$\text{attention\_span} = \max_{q \in Q} \sum_{k \in K} d_{qk} * A_{qk} \tag{3.2}$$

For a given example, we can take the maximum attention span score of all its tokens, and for a model, or a layer of a model, we can report the averaged attention span score over a set of examples. In this section, we compare the average attention span scores of several transformer language models, using raw attention weights, and attention weights post-processed by attention-rollout.

First of all, we observe that the attention-span scores obtained from attention-rollout change more smoothly across layers, compared to what we get from raw attention weights. Based on the results from attention-rollout, the models' attention span converges as we move up across the layers, while the attentions-span scores computed based on raw attention weights seem random and are less reliable as they don't take the mixing of information across the layers of the self-attention layers into account.

In line with our findings from chapter 2, we find that Albert, Roberta and Electra-large are more sensitive to context length, as their attention span scores are higher compared to BERT. Moreover, an interesting observation here that calls for further future investigation is that the attention span of different models changes differently when they are scaled up. In the case of BERT, we observe both BERT-base and BERT-

large converging to the same attention span score in their last layer, while BERT-base converges faster. In Electra increasing model size leads to a higher attention span while for Albert and Roberta, it has a reverse effect. Speculating about the reasons behind these behaviours is beyond the scope of our experiments. It is noteworthy, that none of these differences is observable based on raw attention scores.

3.5 Conclusion

In this chapter, we introduce and explore two techniques, attention rollout and attention flow to approximate attention from any node in any layer of a Transformer model to any other node, given raw attention scores. Our ideas are simple and task/architecture agnostic. We insisted on sticking with simple ideas that only require attention weights and can be easily employed in any task or architecture that uses self-attention.

Translating embedding attentions to token attentions can provide us with better explanations about models' internals. Yet, we should be cautious about our interpretation of these weights, because, we are making many simplifying assumptions when we approximate information flow in a model with the attention weights.

We should note that all our analysis in this chapter is for a Transformer encoder, with no casual masking. Since in the Transformer decoder, future tokens are masked, naturally there is more attention toward initial tokens in the input sequence, and both attention rollout and attention flow will be biased toward these tokens. Hence, to apply these methods to a Transformer decoder, we should first normalize based on the receptive field of attention.

Comparing the attention maps obtained with attention flow and attention rollout with raw attention maps, in a few different scenarios, we show that in all these cases, the attention flow and attention rollout can provide us with more comprehensible explanations of how the information flows through the self-attention layers.

In particular, one thing that we show in this chapter, is that applying methods such as attention rollout and attention flow on raw attention scores, to compute effective attention of input tokens, leads to scores that are more aligned with other input attribution methods such as blank-out compared to raw attention weights. More recently, it has been shown that indeed attention flow can approximate shapely values [Ethayarajh and Jurafsky, 2021, Metzger et al., 2022]. Shapely value is a concept in cooperative game theory that assigns a unique credit to each player proportional to its contribution toward the total gain [Kuhn and Tucker, 1953]. This concept is applied in interpretable machine learning as an input attribution method [Lundberg and Lee, 2017].

Furthermore, following this work, we could build the attention graph with effective attention weights [Brunner et al., 2020] instead of raw attentions. It is also possible to adjust the attention weights using gradient-based attribution methods [Ancona et al., 2019], as is explored by Chefer et al. [2021].

PART

II

From Vectors to Voxels

How do we establish a relationship between computational models of language and data on the human brain activation while they process language? Pioneering work of Mitchell et al. [2008] showed that techniques from distributional semantics could be used to predict and decode brain activations. In the decade since that paper, many efforts have been reported using brain data to evaluate computational models, using NLP models to build predictive models of the human brain, or both [Abnar et al., 2018, Bingel et al., 2016, Bulat et al., 2017, Fyshe et al., 2014, Huth et al., 2016, Murphy et al., 2012, Pereira et al., 2018, Ruan et al., 2016, Søgaard, 2016, Wehbe et al., 2014a, Xu et al., 2016]. In the early stages, most of the work in this area was focused on lexical representations, reporting promising results for concrete nouns, presented in isolation [Pereira et al., 2018]. More recently researchers have tried to adapt the methodology to address words in context, in sentence and story processing tasks. Pereira et al. [2018], for instance, used a bag of words model of sentence meaning to decode sentences from brain activation. Qian et al. [2016], Wehbe et al. [2014b] use the internal states of LSTMs trained for language modelling for predicting brain signals. Jain and Huth [2018] report that the higher layers of the LSTM are better at predicting the activation of brain regions that are known for higher-level language functions (a finding seemingly at odds with results in chapter 5).

The main building blocks of these approaches are (1) a computational model to provide representations of language at the level of interest (words, phrases, sentences or stories); (2) neurological signals to provide representations of language at the level of interest; (3) a method to quantify the correlation between the brain signals and the representations obtained from the computational models. These frameworks, on the one hand, enable measuring the cognitive plausibility of different language models, i.e., the similarity of the solution learned by a model to the solution represented in a human subject's brain. On the other hand, they allow using computational models as a bridge to connect linguistic theories to empirical evidence based on biological signals. The latter is complementary to direct probing of brain data and seems to be vital since in many cases it is not even possible to directly probe the brain activity patterns for a certain phenomenon since they are not easy to formalize as features that can be decoded from brain signals. For instance, when we want to probe the processes that are involved in forming the final representations. E.g., if we want to validate the architectural inductive biases of neural network language models such as recurrence.

In this part, in chapter 4, we use the dataset introduced by Mitchell et al. [2008] to evaluate context-independent word representation models regarding their similarity with brain activity patterns. By comparing word embedding models trained with different supervision signals, we aim to understand what type of information is encoded in brain activity patterns of human subjects processing isolated nouns. In chapter 5, we try to extend this framework a step further, and use neural network-based language models, with different architectures and trained with different language modelling objectives, as explanatory models for brain activity patterns of human subjects reading a story.

In our efforts, we run into several major conceptual, methodological and technical

challenges. Most importantly: how do we determine what we are really observing in the brain data? Are we really seeing signatures of linguistic processes, or just neural correlates of general cognitive processes evoked by a correct understanding of the linguistic input? How do we adequately control for alternative explanations of the observed correlations? And how do we deal with the intricate temporal dynamics and the overwhelmingly high dimensionality of the brain, and the very indirect, delayed and/or coarse measurements that neuroimaging gives us of the processes in the brain? Merely demonstrating a correlation between two black boxes is clearly not sufficient. We argue that experiments to find the model best correlated with brain activations should be accompanied by efforts for interpreting the internal representations and operations of the models.

Additionally, it appears that, when comparing a set of computational models in terms of their explanatory power of the neurological data, choices for the type of brain signals and the method for computing the correlation can significantly impact the conclusions [Beinborn et al., 2019]. This adds to the challenges of research on this topic and calls for efforts toward building more robust and standardized approaches for this purpose.

4

Word Representations: Machines and Brains

How may different computational word representation models help in understanding how words are represented in the human brain? Can these models help us capture different properties and features reflected in human brain neural activation patterns when processing words? Mitchell et al. [2008] pioneered the use of corpus-derived word representations to predict patterns of neural activations when subjects are exposed to a stimulus word. Using the same approach, in this chapter, we present a systematic evaluation of different word embedding models, against the neuroimaging data from [Mitchell et al., 2008]. We compare neural word embedding models with traditional approaches that are based on manually assigned linguistic word attributes, and neuro-

This chapter is based on the following paper.

- Samira Abnar, Rasyan Ahmed, Max Mijnheer, and Willem Zuidema. 2018. Experiential, Distributional and Dependency-based Word Embeddings have Complementary Roles in Decoding Brain Activity. In Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018), pages 57–66, Salt Lake City, Utah. Association for Computational Linguistics.

- List of contributions is as follows. Samira Abnar: Preparing the framework for running the experiments, Designing and Running experiments, and Writing the paper. Rasyan Ahmed: Running preliminary experiments for investigating the correlation between brain activity patterns and general purpose word embedding models [Ahmed, 2017]. Max Mijnheer: Came up with the idea of using experiential word embeddings, Ran the preliminary experiments for investigating the correlation between brain activity patterns and experiential word embeddings [Mijnheer, 2017]. Willem Zuidema: Planning and guiding the research, Reviewing and revising the paper.

inspired techniques based on sensory-motor features. Besides a large-scale evaluation of various word embedding models, we conduct a detailed error analysis to understand the differences between them. Our results suggest that for studying properties of brain signals, we can avoid using word representation models specifically designed for every study. Instead, we can move towards using general-purpose word embedding models that excel at a variety of NLP tasks.

4.1 Evaluating Word Representation Models with Brain Imaging Data

How are word meanings represented in the human brain? Is there a single amodal semantic system or are there multiple responsible for representing meanings of different classes of words? A series of studies have shown that a combination of methods from machine learning, computational linguistics and cognitive neuroscience are useful for addressing such questions.

Mitchell et al. [2008] pioneered the use of corpus-derived word representations to predict patterns of neural activations when subjects are exposed to a stimulus word. Using their framework, a series of papers have evaluated various techniques of computing word representation models based on different assumptions. Later, Huth et al. [2012] extends these findings by studying fMRI responses collected from human subjects watching a movie. Still using hand-crafted feature vectors based on WordNet [Miller, 1995], they study whether a continuous semantic space underlies category representation in the human brain.

Since these early successes, a range of new word embedding methods have been proposed and successfully used in a variety of NLP tasks, including methods based on neural networks. Baroni et al. [2014] and Pereira et al. [2016] present systematic studies, showing that also behavioural data from psycho-linguistics can be modelled effectively using a general-purpose neural word embedding models such as GloVe[Pennington et al., 2014] and word2vec[Mikolov et al., 2013]. At the same time, studies in the area of vision have shown that deep learning models fit very well to the neocortical data [Cadieu et al., 2014, Khaligh-Razavi and Kriegeskorte, 2014] and they can help to better understand the sensory cortical system [Yamins and DiCarlo, 2016].

To investigate how well the new word embedding models, and in particular the deep learning models, fare in helping to understand neural activation patterns in the domain of language, we present a systematic evaluation of different word embedding models, against the neuroimaging data from [Mitchell et al., 2008], following the experiments and preliminary results in [Ahmed, 2017, Mijnheer, 2017].

Our main goal is to evaluate the usefulness of these different word embedding models for understanding different properties and features reflected in human brain neural

activation patterns. To address this goal, we take word embedding models based on different assumptions of how meanings of words can be represented, and evaluate their performance on either the task of predicting brain data from word embeddings or the reverse, predicting word embeddings from brain data. The basic assumption here is that the better the performance of a model is the more probable it is that the way the word embedding model is built reflects what happens in the human brain to understand a meaning of a word. In our experiments, we compare neural word embedding models with traditional approaches that are based on manually assigned linguistic word attributes, and neuro-inspired techniques based on sensory-motor features. Besides a large-scale evaluation of various word embedding models, we conduct a detailed error analysis to understand the differences between them.

The first research question we investigate is: How well does each word representation model correlate with neural activation patterns in the human brain? To answer this we measure how well different word embedding models can predict the brain imaging data. Taking this one step further, we also train our models in the reverse direction: to directly predict word embeddings from brain data.

The second research question that we investigate is: What is the best word embedding model for predicting brain activation for different classes of nouns? Maybe the human brain uses different processes to understand the meanings of different kinds of words [Caramazza et al., 1990, Caramazza and Shelton, 1998, Riddoch et al., 1988, Warrington and Shallice, 1984]. We do a qualitative analysis of our results to see whether different word embedding models are good at predicting brain activation for different categories of nouns. The third question we address is Which are the most predictable voxels in the brain for each word embedding model? By answering this question we want to test the hypothesis that different areas of the brain are responsible for processing different aspects of the meaning of nouns. If different models have different performances either for different noun pairs or for different brain areas, the next step would be to find a way to integrate different models to build a model that better fits the brain data.

Two main approaches are proposed for evaluating word representations with brain imaging data. In the first approach, the computational representations of words are employed to predict the neural word representations, i.e., the neural activation patterns [Mitchell et al., 2008]. The second approach is based on the correlation between the pairwise similarity of word representations in the brain and the computational model under evaluation. [Anderson et al., 2016]. An extensive overview of these techniques is provided in chapter 2.

Here, we focus on the Brain Activation Prediction Task. Given a word, we want to predict how the brain activation for a particular person would look like if he/she is exposed to the noun and is trying to understand its meaning. We train a linear regression model to map the embedding of each word to its corresponding fMRI scan. Then we test how well the model can discriminate between two unseen examples. The brain activation is represented as an n-dimensional vector which each element of it is a

projection of a tiny space in the brain, called a voxel. Each word is also represented as an n-dimensional vector which its interpretation would depend on the kind of word embedding model that is being used. The subgoal here is to find a function that computes brain activation from word vectors or vice versa.

If we can train a linear regression model that successfully predicts brain activation patterns from word representations for the words that are not in its training set, it means that there is a mapping between the computational word embedding in use and the word representation in the brain. This indicates that the information encoded in the computational word representation model is enough to reconstruct brain activation patterns. On the other hand, if for a word embedding model, we can train a regression model to predict word representations from brain activation patterns, this would mean that the information needed to compute the word representation is to some extent encoded in the neural activation patterns.

Experimental Setup As the regression model, we employ a single-layer MLP with *tanh* activation. To avoid over-fitting we use drop-connect [Wan et al., 2013] with a keeping rate of 0.7 beside L2 regularization with $\lambda = 0.001$. In all the experiments we train the models for each subject separately. The training and evaluation are done with the leave-2-out method as suggested in [Mitchell et al., 2008]. Where we train the model on all except 2 pairs and then evaluate the performance of the model on the left-out pairs. We do this for all possible combinations of pairs.

Neuroimaging Data Our experiments are conducted on the data from Mitchell et al. [2008] which is publicly available[1]. This is a collection of fMRI data that is gathered from 9 participants while exposed to distinctive stimuli. The stimuli consisted of 60 nouns and corresponding line drawings. Each stimulus was displayed six times for 3 seconds in random order, adding to a total of 360 fMRI images per participant.

Word Embedding Models In order to get insights about how the human mental lexicon is built, we use a wide variety of word representation models. The word embedding models that we are exploring in our experiments are in two (non-exclusive) categories: experiential and distributional. In the experiential model, the meanings of the words are coded to reflect how the corresponding concept is experienced by humans through their senses. In the distributional models, the meaning of words is represented based on their co-occurrence with other words. These models can be either count-based or predictive [Baroni et al., 2014]. The word representation models we will use are:

- **Experiential word representations**: Experiential word representations are suggested based on the fact that humans remember the meaning of things as they experience

[1] *http://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html*

them. In [Binder et al., 2016] a set of 65 features are defined and crowdsourcing is used to rate the relatedness of each feature for each word. Thus, instead of computing the value of features using statistical data from textual corpora, they use actual human ratings. We use the dataset introduced in [Binder et al., 2016]. Since it contains only about 50% of the nouns in the Tom Mitchell et al dataset, some of the experiments we report are with this limited noun set.

- **Distributional word embedding models**:
  - **Word2Vec**: Word2vec basically is a shallow, two-layer, neural network that reconstructs the context of a given word. In our experiments, we use the skip-gram word2vec model trained on Wikipedia [Mikolov et al., 2013].
  - **Fasttext**: Fasttext is a modification of word2vec that takes morphological information into account [Bojanowski et al., 2016].
  - **Dependency-based word2vec**: The dependency-based word2vec introduced in [Levy and Goldberg, 2014] is a word2vec model in which the context of the words is computed based on the dependency relations.
  - **GloVe**: GloVe is a count-based method. It does a dimensionality reduction on the co-occurrence matrix[Pennington et al., 2014].
  - **LexVec**: LexVec is also a count-based method. It is a matrix factorization method that combines ideas from different models. It minimizes the reconstruction loss function that weights frequent co-occurrences heavily while taking into account negative co-occurrence [Salle et al., 2016a,b].
- **25 verb features**: Similar to experiential word representations, this model is based on the idea that the neural representation of nouns is grounded in sensory-motor features. They have manually picked 25 verbs and suggested using the co-occurrence counts of nouns with these 25 verbs to form the word representations [Mitchell et al., 2008].
- **Non-distributional word vector representation**: [Faruqui and Dyer, 2015] have constructed a non-distributional word representation model employing linguistic resources such as WordNet[Miller, 1995], FrameNet[Baker et al., 1998] etc. In this model, words are presented as binary vectors where each element of the vector indicates whether the represented word has or does not have a specific feature. As a result, the vectors are highly sparse. The advantage of this model to distributional word representations is the interpretability of its dimensions.

4.2 Correlation Between Word Embeddings and Brain Activation Patterns

To understand how relevant are the information captured in brain signals and word embedding models, we investigate the predictability of brain signals from word repre-

sentations and vice versa.

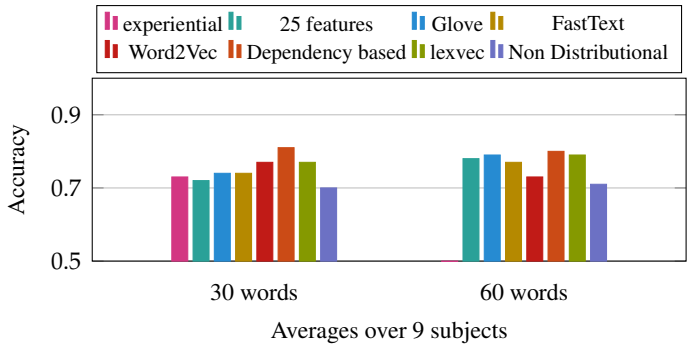


Figure 4.1: Results for the word-to-brain activation prediction task (Chance is .5).

4.2.1 Predicting Brain Activation for Noun Stimuli

First, we train a separate regression model for each word representation model to compute the average brain activation corresponding to each word for a particular subject. Figure 4.1 illustrates the results of evaluating these models on the brain activation prediction task, using the leave-2-out methodology, as discussed above. For the sake of including the experiential word representations from [Binder et al., 2016] in our evaluations, we also conducted a set of experiments with only the nouns that were included in the experiential word representation collection. The good news is that all the models we are evaluating perform significantly above chance. The fact that the ranking of the models differs per subject makes it difficult to make general conclusions about the best model. Overall, dependency-based word2vec, GloVe and 25 features model are the top-ranked models for at least one of the subjects.

Among neural word embedding models, dependency-based word2vec is achieving the best accuracy (81%). This is in line with the results from [Murphy et al., 2012], where they showed that the corpus-based model considering the dependency relationships has

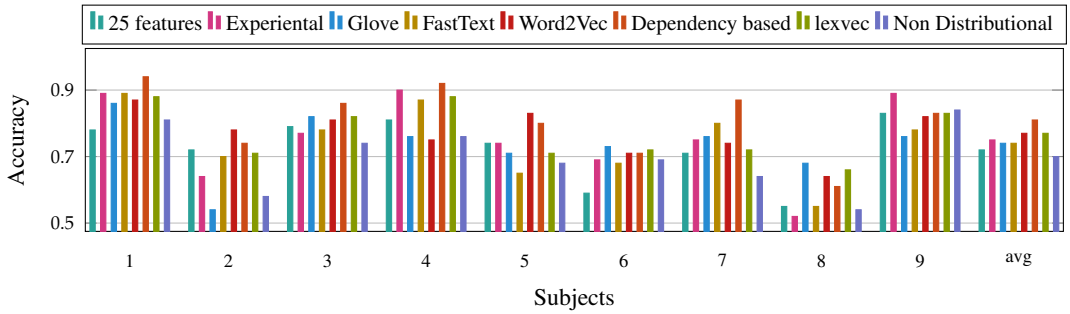


Figure 4.2: Results of different word representation models for the word-to-brain activation prediction task for the limited set of words, split per subject.

the highest performance among corpus-based models. These authors report an accuracy of 83% (with 1000-dimensional word vectors). Somewhat higher still than the best dependency-based word2vec, and the highest performance reported in the literature until now for a corpus-based model. Fasttext and dependency-based word2vec are performing better than word2vec; this might reflect the importance of morphological and dependency information. Comparing predictive models with count-based models, although count-based methods like GloVe and LexVec are beating simple word2vec, looking at the performances of fasttext and dependency-based word2vec, we can conclude that the context prediction models can potentially perform better.

Moreover, comparing the performance of the Experiential Model with the 25-feature model, we see that the Experiential Model is doing slightly better on average while their ranking is different per subject. Either the higher number of features or the way feature values are computed could have led to a slight improvement in accuracy for the experiential model.

In both sets of experiments in Figure 4.1 the non-distributional word representation model has the lowest performance. The very high dimensionality of the brain imaging data versus the sparseness of non-distributional word vectors makes training the regression model with these vectors much harder and this might be the primary reason for its low performance.

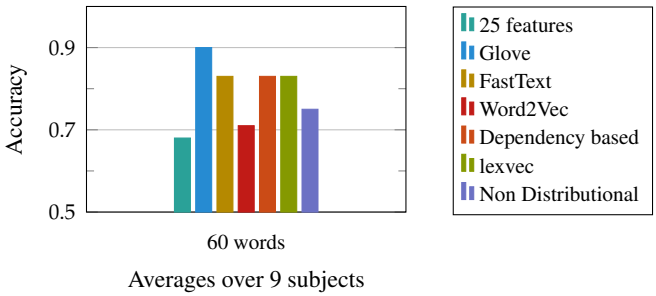
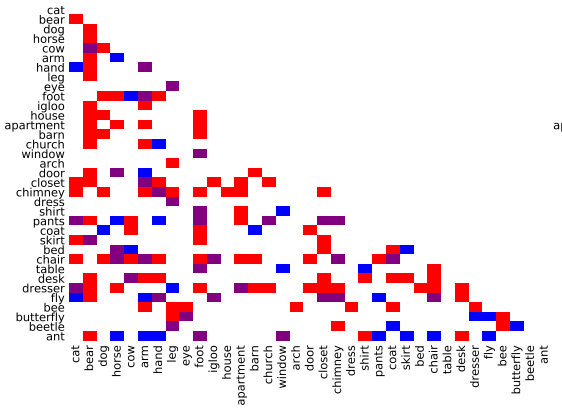


Figure 4.3: Results of different word representation models for the brain activation to word representation prediction task.

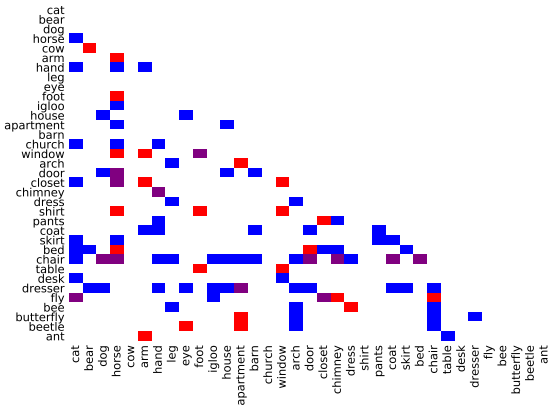
4.2.2 Decoding Words from Brain Activations

Next, instead of predicting brain activation patterns, we train the regression model to predict the word representation given a brain activation. Thus, we want to predict the stimulus word from the neural activation pattern in the brain. Evaluation is still based on the leave-2-out setup (so we still evaluate with two brain images and two word embeddings at each instance, making quantitative results comparable across experiments). The results are shown in Figure 4.3. We expected the performance of the models on the reversed task, predicting word features from brain activation, to be

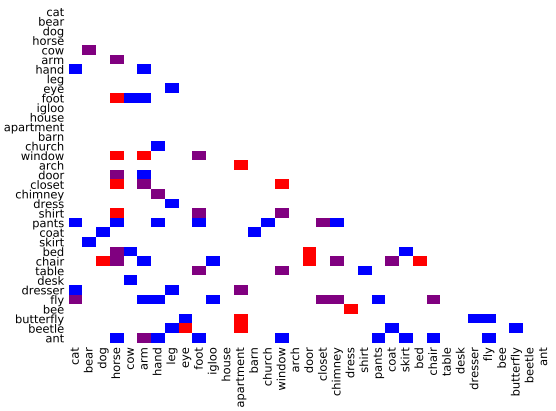
somewhat similar to their performance on the main task, predicting brain activation patterns from word vectors.



(a) 25 features model (red) vs experiential model (blue)



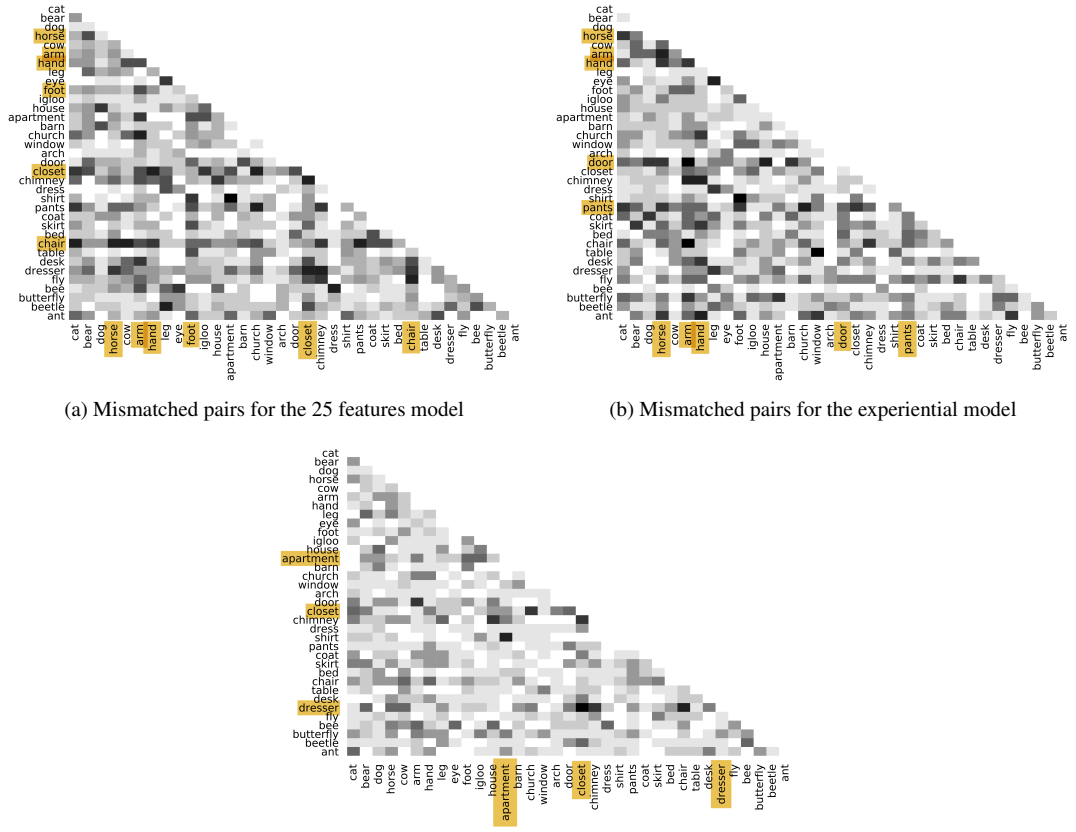
(b) dependency based word2vec (red) vs word2vec (blue)



(c) dependency based word2vec (red) vs experiential model (blue)

Figure 4.4: Comparing different models mismatched pairs for subject 1 (purple indicates word pairs confused by both models).

However, the results are surprising. For the 25 features model, the accuracy on the reversed task is much lower. This may be because of the way the feature vector for nouns is distributed in the space in this model. Or it could be that neural activation patterns do not encode all the necessary information to approximate these feature values. This could indicate that while the 25 features model is pretty useful in interpreting brain activation patterns it is not a plausible model to simulate how nouns are represented in the human brain. On the other hand, it seems that it is very easy to construct GloVe word vectors from brain activation patterns; this model achieves an accuracy of 90%. In [Sudre et al., 2012] accuracy of 91.19% is reported on a similar task on MEG data. GloVe is based on the distributional semantics hypothesis, and it is achieved by learning to predict the global co-occurrence statistics of words in a corpus. Hence, obtaining high accuracy in the word prediction task using GloVe, supports the fact that the context of the words has a major role in the way we learn their meanings. The important thing



(a) Mismatched pairs for the 25 features model

(b) Mismatched pairs for the experiential model

(c) Difference between mismatched pairs

Figure 4.5: Comparing mismatched pairs for the 25 features model and the experiential model averaged over all subjects.

to notice is that of course the more information we encode in the word representation the more powerful it becomes in predicting neural activation patterns as far as that information is relevant to some extent. However, this alone doesn't imply that the exact information is encoded in the neural activation patterns. As we can see in our results, compared to GloVe, it's not that easy to reconstruct the Fasttext and dependency-based word vectors from the brain activation patterns. We conjecture that while morphological and dependency information is helpful in learning word representations that are to some extent more similar to the neural representation of nouns in our brain, this information is not explicitly encoded in the brain activation patterns.

In the end, only comparing the accuracy of these models does not reveal much about the differences between them and does not mean that the model with the highest accuracy can replace all the others.

4.3 Best Word Embedding for Predicting Brain Activations of Different Nouns

In order to get more insights about the differences between the models, we look into the errors they make. It is informative to see whether each of these models is good at predicting neural activation patterns for a different group of noun pairs. We want to test the hypothesis of whether the human brain uses different mechanisms for understanding meanings of different categories of words [Caramazza et al., 1990, Caramazza and Shelton, 1998, Riddoch et al., 1988, Warrington and Shallice, 1984]. To investigate this we look into the miss-matched noun pairs for each of the word representation models. We want to see which are the most confusing noun pairs for each model and measure the overlap between the errors the models make. This will reveal if these models are actually encoding different kinds of information.

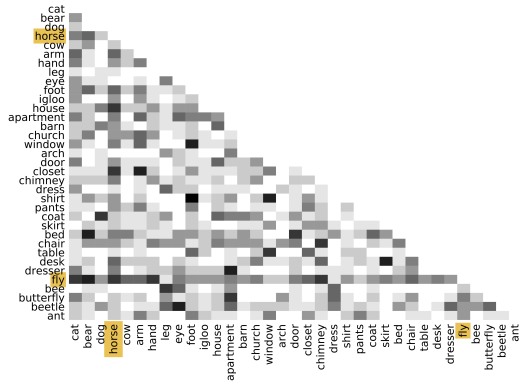
Figures 4.4a, 4.4c and 4.4b show the overlap between mismatched pairs for different models for subject 1. In these plots, the red color corresponds to the first model mentioned in the caption, the blue colour corresponds to the second model and the purple colour indicates the overlaps. While there is some overlap between the mistakes of the 25 features model and the experiential model, a considerable number of mismatched pairs are not in common between them.

One interesting fact about the 25 features model is that for some specific nouns ie. “bear”, “foot”, “chair”, and “dresser”, no matter what is its pair, discrimination performance is poor. eg. “bear” is not only confused with other animals, but also with some body parts, places, etc. We do not notice similar phenomena for the experiential model. This could be a side effect of using co-occurrence statistics from corpora to learn word representations and could show that for some reason the representations learned for these nouns are not distinguishable from other nouns.

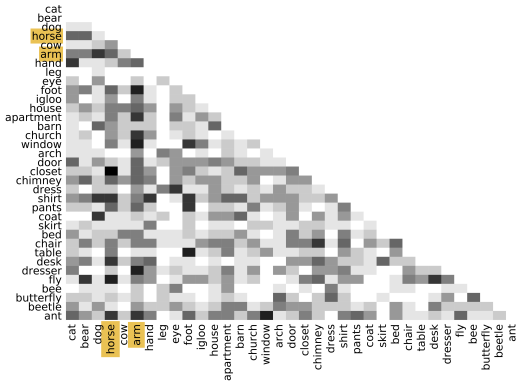
Looking into the noun pair mismatches of the experiential model and the dependency-based word2vec in Figure 4.4c, again we see a considerable amount of overlap. They both perform equally for discriminating among animals. But the experiential model makes more mistakes about “body parts” and “insects”. Comparing the dependency-based word2vec with simple word2vec, in Figure 4.4b we observe similar patterns to Figure 4.4a. As illustrated in the plot, discriminating some words, eg., “chair” is difficult for word2vec while it’s not the case for dependency-based word2vec. It seems like both experiential attributes of nouns and the dependency information is helping in learning more distinguishable representations for nouns.

4.3.1 25 Features vs Experiential

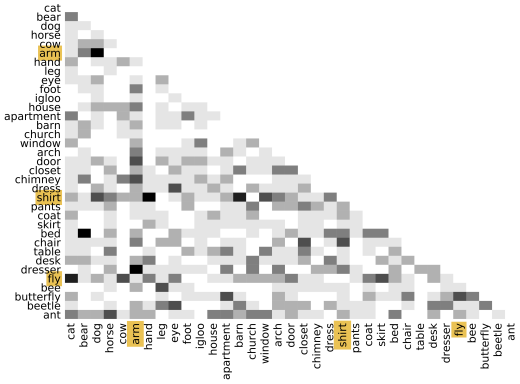
As shown in Figure4.1, the experiential model performs better than the 25 features model on average. Considering the fact that these two models are reflecting the same



(a) Mismatched pairs for dependency-based word2vec



(b) Mismatched pairs for GloVe



(c) Difference between mismatched pairs

Figure 4.6: Comparing mismatched pairs for dependency based word2vec and GloVe averaged over all subjects

underlying theory, we might expect that if one of them is more accurate, it can replace the other. However, by looking into the difference between their mismatched pair, Figure 4.5, we observe that the mistakes these two models make are not completely overlapping: the nouns ‘arm’ and ‘hand’ are difficult to discriminate for both models, while ‘chair’ and ‘house’ are among the nouns with most mistakes for the 25 features model, and ‘horse’ and ‘door’ for the experiential model. For both models, most mismatches are in the category of body parts.

4.3.2 GloVe vs Dependency-based Word2vec

We also compare the mismatch pairs for GloVe and dependency-based word2vec as the two neural models that achieve the highest accuracies in Figure 4.6. These two models are different both in the richness of the information they use to learn word representations, and also in the way they use this information. In GloVe, the model is trained based on the global co-occurrence of words whereas in word2vec word representations are learned based on the context of the words for each example locally. For GloVe, similar to the 25 features model and the experiential model, ‘arm’ is one

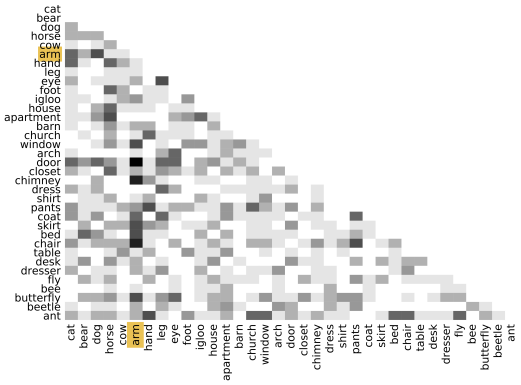


Figure 4.7: Difference of mismatched pairs for dependency based word2vec and experiential model

of the hardest-to-discriminate nouns. But the 'body parts' category is not as confusing as for the experience-based models. For the dependency-based word2vec, the patterns of errors are somehow different and the most difficult word seems to be 'fly'. This is because 'fly' can be either a verb or a noun, and since it is more frequent as a verb, the dependency-based model is learning the representation of its verb form. For GloVe, this is not very problematic because it is only based on co-occurrence counts, thus, an average representation is learned. In general, despite the fact that these two models are based on different assumptions their mismatches have more overlap than for the two experiential models. This may be a side effect of the fact that they both make fewer mistakes.

4.3.3 Experiential vs Dependency-based Word2vec

The mismatched pairs of the experiential model and the dependency-based word2vec and their differences are illustrated in Figure 4.7. The experiential model seems to have less prediction accuracy for noun pairs in the same category.

4.4 Most Predictable Voxels in the Brain Given Different Word Embeddings

Each of the computational models of word representation we have employed to predict brain data is based on modelling different aspects of words' meanings. Now we want to investigate if our brain is doing a combination of all these mechanisms and if different groups of voxels in the brain are responsible for processing each aspect. One way to test this is to look into the predictability of different voxels with each of these models. For this purpose, we have identified the top 50 most predictable voxels for each model. In Figure 4.8 you can see the 50 most predictable voxels for dependency-based word2vec

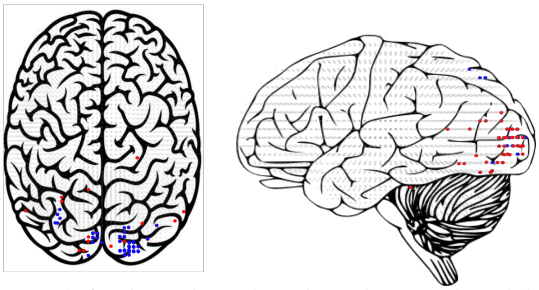


Figure 4.8: Most predictable voxels for dependency based word2vec(red) and the experiential model(blue)

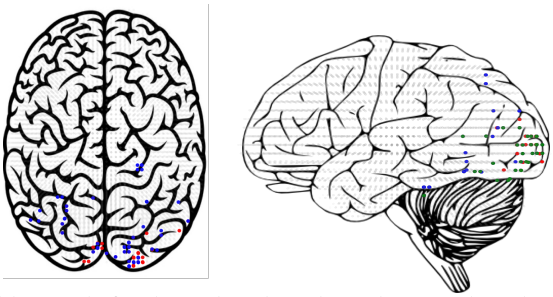


Figure 4.9: Most predictable voxels for dependecy based word2vec(red) and word2vec(blue). Green dots are among the top 50 voxels of both models.

and the experiential model. In Figure 4.9 you can see the 50 most predictable voxels for dependency-based word2vec and simple word2vec. The green colour indicates the common top voxels between the two models. From these figures, we can see that there is a lot more overlap between the dependency-based word2vec and word2vec, compared to the experiential model.

A Mixed Model If each model is good at predicting the neural activation pattern for a different group of nouns/different groups of voxels, theoretically, it is possible to build a better model using an integrated model. In other words, we should be able to improve the accuracy of predicting neural activation patterns by employing a combined model. We conduct a new experiment by integrating the dependency-based word2vec as a neural corpus-based word representation with the experience-based models, i.e., the 25 verbs model and the experiential model. We expect the performance of the model to be a little bit higher than the dependency-based word2vec. Our results indicate that combining the dependency-based word2vec with the experiential model linearly doesn't lead to an improvement in the accuracy over the limited set of words available in the experiential model. However, linearly combining the 25-feature model with the dependency-based word2vec leads to an accuracy of 82% over the 60 nouns, which is 2% higher than the accuracy of the dependency-based model.

4.5 Conclusion

Based on our systematic comparison, we can conclude that the deep learning models for learning word representations fit well with brain imaging data for nouns. Models like dependency-based word2vec are already beating the experiential word representation models that were particularly designed for brain activation decoding tasks. These findings suggest that in future studies for understanding what is captured in the neural responses of human subjects when they are processing language, we can rely on general-purpose word representation models. Thus, avoiding the need to design word representation models specific to each study. Not only these general-purpose models are shown to be successful at solving a variety of NLP tasks, but they also have a better predictive power against brain signals compared to hand-crafted features.

Moreover, focusing on the methodology, comparing the results of learning the mappings from words to brain activations and vice versa, convinces us that it is important to study the performance of the models in both directions to really understand what kind of information is encoded in the neural activation patterns for words.

Looking into the details of the performance of the models we study, it turns out that each of them makes different kinds of mistakes. We speculate that this difference in performance explains that these models capture different aspects of words' meanings. To further investigate this, we build a model that combines the experience-based word representation model with the dependency-based word2vec. By linearly combining the 25 features model with the dependency-based model we are able to achieve a higher accuracy on the brain activation prediction task.

Our results suggest that, even for the simple case of computing representations of nouns, there is still room for improvement. We need more advanced models that can capture different aspects of the meaning of the words. For example, one of the main problems of the corpus-based distributional models that we have applied is that they do not account for different senses of the words. Hence, the representations they learn for words with more than one sense can be noisy and biased toward the most frequent sense.

Additionally, words in isolation as we studied in this chapter are not representative of language processing in more natural setups. A more promising approach would be to study the brain signals of humans when processing linguistic inputs in context. For example, we can study the representations of words or phrases in the context of a story or dialogue. On the computational side, we need models that can account for the dynamic context of the stimuli. For example, the representation of a word with multiple senses out of context should be different than its representation when there is some context to disambiguate its meaning. In the next chapter, we will make an effort to do this, where we employ language models that can potentially capture the effect of context on the word representations [Jozefowicz et al., 2016, Peters et al., 2018b] to investigate neural responses collected from human subjects during a story reading task [Wehbe et al., 2014a].

5

In Search of Footprints of the Story in Language Models and Brain Activations

Recently, pretrained language models are shown to be very effective in learning useful contextualized representations [Devlin et al., 2019, Peters et al., 2018b]. Employing the representations obtained from these models we can achieve impressive performance, in some cases at or beyond the human level, on a variety of NLP tasks. Are these models learning language similar to how the human brain processes language? Or are they converging to completely different types of solutions? Are the factors in designing and training these models that lead to improvements in their performance, cognitively relevant? Do these models learn and process information in a similar manner to humans? To answer these questions, we investigate whether different neural networks trained on

This chapter is partially based on the following paper (some of the contents are not published before).

- Samira Abnar, Lisa Beinborn, Rochelle Choenni, and Willem Zuidema. 2019. Blackbox Meets Blackbox: Representational Similarity & Stability Analysis of Neural Language Models and Brains. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 191–203, Florence, Italy. Association for Computational Linguistics.
- List of contributions is as follows. Samira Abnar: Designing and running the experiments, Writing the paper. Lisa Beinborn: Contributed to the discussion about the paper. Helped in revising the earlier versions of the paper. Rochelle Choenni: Contributed to the discussion about the paper. Helped with some of the visualizations. Willem Zuidema: Guiding the research, Writing the paper.

language modeling can explain human brain activations during a story reading task.

5.1 Language Models and Brain Activity Patterns in the Context of a Story

When we read a story, we perceive and comprehend information in a sequence. Our mental state gets updated step by step as we read words one by one. We use the previous words to understand the meaning of the current word and update our understanding of the previous content as we move forward in the story. Additionally, we predict the upcoming content, and this too affects how we understand the current content.

Recently, pretrained language models are shown to be very effective in learning useful contextualized representations [Devlin et al., 2019, Peters et al., 2018b]. Employing these representations we can achieve state-of-the-art performance on a variety of NLP tasks. While the ability of these models to capture context is constrained (e.g., with respect to context length), their performance relies on the quality of the context and their ability to capture that [Khandelwal et al., 2018]. On the other hand, brain decoding studies have shown that even using simple RNN-based language models to compute context-aware representations for words are better in explaining brain activations patterns than isolated word representations [Jain and Huth, 2018, Qian et al., 2016]. As hard as it is to believe a model can learn only by reading pure text, i.e. no grounding, and only with a language modelling objective, these studies show that, in fact, language models are learning some useful insight about language, and there is a growing effort to look into these models and understand what kind of knowledge they capture from reading text.

In this chapter, we propose putting these efforts together to answer the question of what can these models teach us about language processing in the human brain. We investigate whether a neural network trained on language modeling can explain human brain activations during a story reading task Our goal is similar to [Qian et al., 2016]. In addition, compared to [Jain and Huth, 2018], our aim is not only to show that using these contextualized word representations we can better explain brain activations, but to reveal why this is the case.

To do so, we use a dataset of brain scans of human subjects during a story reading process. We feed the trained language model the same stimuli that were presented to the human subjects. Then, we study the connection between the internal state of the model and the brain activation patterns. We ask: (1) Can we find a mapping between the information encoded in different parts of the internal state of the language model and the brain imaging data? (2) Do the brain activity data and the internal states of the language model capture similar linguistic features of the story? If different layers of the language model explain brain activation patterns differently, is this because different types of features are captured on different layers? And: (3) do the representations that

are aware of the dynamic context and static word embeddings differ in the regions of the brain for which they make the most accurate predictions?

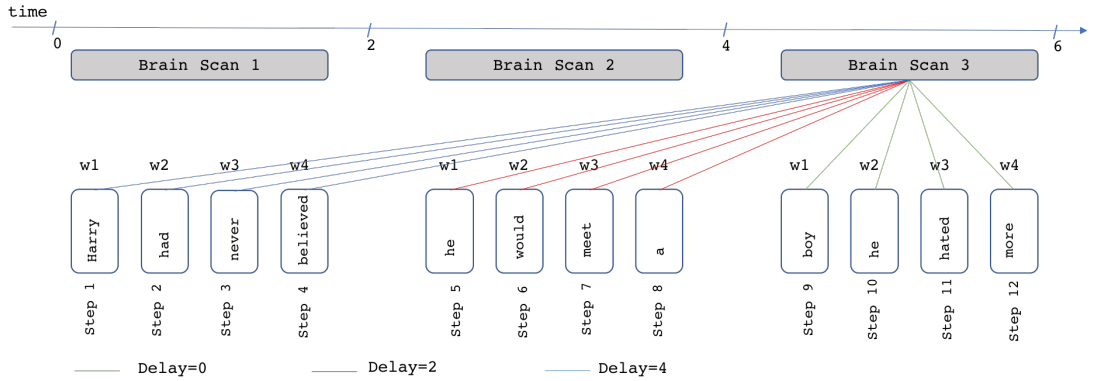


Figure 5.1: Alignment of the words in the story and the brain vectors. Each fMRI scan lasts for 2 seconds during which the subject is reading four words sequentially. Delay is the amount of time in seconds between the time the first of the four words is shown to the subject and when the fMRI scan is started to be taken.

Experimental Setup We compare the representations obtained from different layers of language models to human brain activations captured while reading a story. We use the dataset by [Wehbe et al., 2014a] which consists of the fMRI scans of 8 participants reading chapter 9 of *Harry Potter and the Sorcerer's stone* [Rowling, 1998].[1] For creating this dataset, the story is presented to the participants word by word on a screen in four continuous blocks.[2] Each word is displayed for 0.5 seconds and an fMRI scan was taken every 2 seconds. Figure 5.1 visualizes an example for the beginning of the chapter. More detailed statistical information about the stimuli can be found in Table 5.1.

In our comparisons, we study language models with different architectures trained with different objective functions (see Table 2.1). As a word-level embedding model, we

Table 5.1: Statistics of the Harry Potter dataset.

Block	Words	Unique words	Sentences	Sent Length	Scans
1	1583	553	115	11	326
2	1711	560	163	8	338
3	1411	461	134	8	265
4	1853	583	177	8	366

[1]The data is available at *http://www.cs.cmu.edu/~fmri/plosone/*. Further information on the preprocessing steps is described in the supplementary material.

[2]The story chapter is split into four almost equal-length blocks, each reflecting approximately 12 minutes of measurements. Each block is presented to the participant in one continuous trial, and experimental blocks are separated by pauses for the subjects.

use GloVe [Pennington et al., 2014]. We consider a sentence as a bag of words and take the average of the GloVe embeddings. We employ two LSTM-based language models: ELMO [Peters et al., 2018b] and GoogleLM [Jozefowicz et al., 2016]. Both of these models have two LSTM layers; however, ELMO uses bidirectional LSTM layers, whereas in the GoogleLM the LSTM layers are uni-directional. From these models, we take the internal states of each of the LSTM layers as two different representation spaces. Furthermore, we use BERT and the Universal Sentence Encoder (UniSentEnc), as Transformer based models. BERT is trained on masked language modelling and next sentence prediction tasks [Devlin et al., 2019] while the Universal Sentence Encoder is trained on a different objective than language modelling. The parameters of this model are optimized with respect to different language tasks such that it can better encode the meaning of complete sentences. These two models do not have the recurrent inductive bias of LSTMs, and hence the representations they learn can be completely different.

5.2 Brain Data

The fMRI data contains activation values for approximately 40,000 voxels per scan, each reflecting the oxygen usage (the "BOLD response") in approximately $3mm^3$ of brain tissue. To obtain the brain representations, we flatten the 3D fMRI images into vectors thereby ignoring the spatial relationships between the voxels. We do this either for the whole brain, or for specific regions separately. Not all of the scanned voxels are related to language processing, but the changes in activity might be associated with other cognitive processes like, for example, the noise perception in the scanner. A common reduction method is to restrict the brain response to voxels that fall within a pre-selected set of regions.

In our analysis, we only include the voxels from the top k regions that are most similar across different subjects given the same stimuli. We heuristically set the value of k to 16 based on the distribution of the similarity scores.[3]

Delay An important point to consider when dealing with fMRI data is the hemodynamic response delay: from the time neurons start firing, it takes 4 to 6 seconds until the Bold response reaches its peak [Buckner, 1998]. This means that from the time a stimulus is presented to a subject, it takes approximately 5 seconds before we can observe its response in the fMRI scan of the brain. We account for this delay by varying the alignment between stimuli and scans. If we apply a delay of 0 seconds, scan 3 in the example would be applied to the sequence *boy he hated more*, Figure 5.1. With a delay of 2 seconds, it is aligned to the previous stimulus *he would meet a* and a delay of 4 would result in alignment with *Harry had never believed*.

[3]We sort the brain regions based on their cross-subject similarities for different stimuli and pick a threshold value based on where is a relatively big jump in the similarity scores.

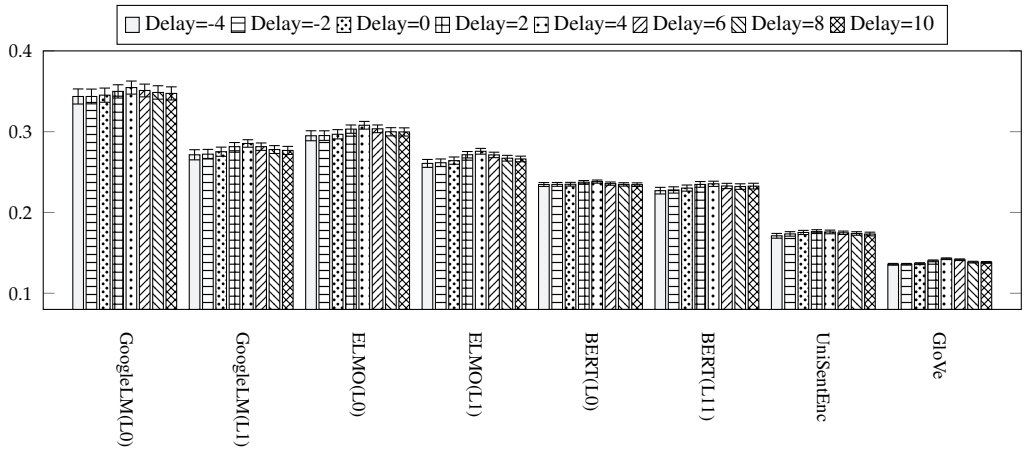


Figure 5.2: Representational similarity of the models and brains averaged over all subjects and the four blocks at different time delays after the human subjects have read the target words, when the context provided to the models is three sentences. Here the delay is increasing from left to right and the error bars indicate the standard deviation across different blocks.

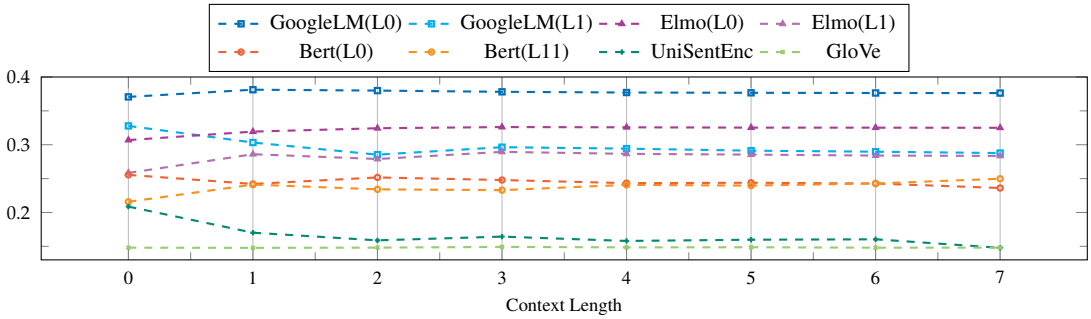
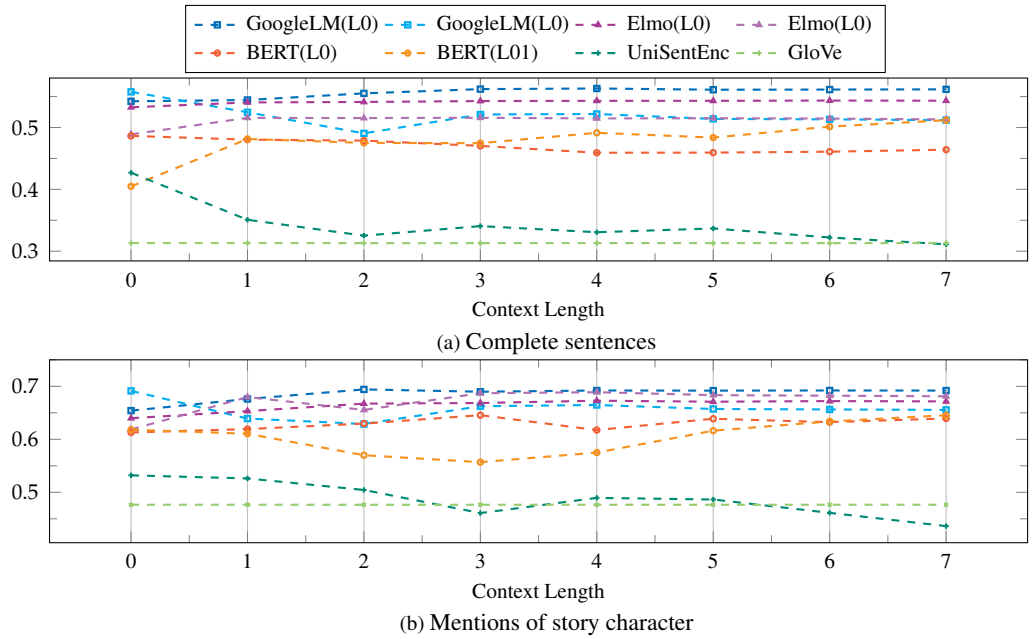


Figure 5.3: Similarity of the representations from different layers of different language models with brain representations given different amount of context, averaged over all subjects. Note that the average representational similarity scores of brains of different human subjects within this data is about 0.55.

5.3 Representational Similarity of Brains and Language Models

In this section, we use standard RSA, as commonly used in neuro-cognitive science. In this approach, in order to compare the representational similarity of two models, given a set of N examples, we first, create the similarity matrices of both models, i.e., a squared symmetrical matrix, $S(M)$, where $S_{ij}(M)$ indicates the similarity of data examples i and j in the representational space of model M. Next, we measure the rank correlation between the similarity matrices of the two models as an indicator of their relational similarity.

Figure 5.2, shows the representational similarity of the models and brains averaged over all subjects and the four blocks at different time delays after the human subjects have read the target words, when the context provided to the models is **3 sentences**. Due to



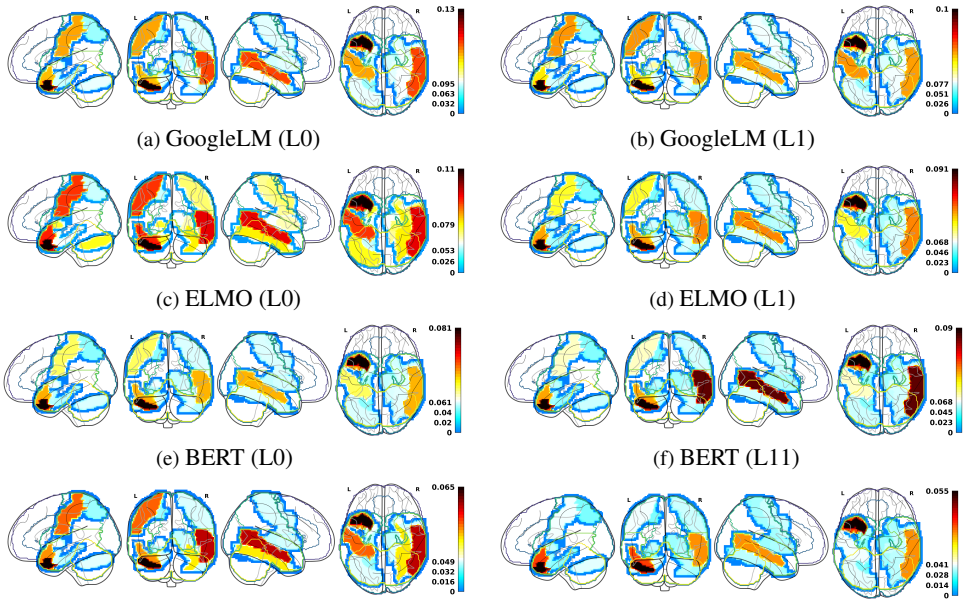
(a) Complete sentences

(b) Mentions of story character

Figure 5.4: Similarity of the computational representations with brain representations at different segments of the story. Note that the average representational similarity scores of brains of different human subjects is around 0.55.

the hemodynamic response delay, we expect to see the peak in similarities after about 4s delay. As we can see, the highest RSA for all models is at $Delay = 4s$, the ranking of the models based on their similarities with brain representations is the same for all amounts of delay. Interestingly, the performances of these models on the NLP tasks are not correlated with their similarity with the brain representations (but note the overall low correlations).

Different Context Lengths Figure 5.3 shows the similarity of different layers of several neural language models with brain signals, with respect to different amounts of context provided to the models, averaged over all human subjects. The representations learned by LSTM-based models are most similar to the brain data, and for both ELMO and GoogleLM the representations from lower layers, $L0$, have higher similarity scores compared to the higher layers, $L1$. Interestingly, for UniSentEnc, BERT($L11$) and also GoogleLM($L1$), increasing the context length, which usually boosts the performance of language encoding models in language understanding tasks [Wang and Cho, 2016], leads to lower similarity with brain representations. It seems that the way these models integrate the context information pushes the representation further away from the brain representations. This could mean: (1) These models are doing fairly well at encoding the local context, but not at a more global level. Alternatively, (2) The information about the more global aspects of the meaning is not encoded in the brain representations.



(a) GoogleLM (L0) (b) GoogleLM (L1)

(c) ELMO (L0) (d) ELMO (L1)

(e) BERT (L0) (f) BERT (L11)

(g) UniSentEnc (h) GloVe

Figure 5.5: Representational similarity of representations learned at different layers of different models with representations at different regions of Subject4's brain which is chosen randomly (the code accompanying this paper can be used to generate the plots for the other subjects). In order to emphasize the difference of the similarity of each model with different brain regions, the color bar is scaled independently for each model. The darkest region for all models is the Left Anterior Temporal Lobe.

Different Segments of the Story If during training, the models are only trained on full sentences, it might be the case that the quality of their representations, when given complete sentences, is significantly better than when provided with incomplete sentences. On the other hand, the representation of sentences in the brain might also be more reliable when the full sentence is read. To take this into account, we look at the similarities of each of the models with brain representations, only at the steps in the story where an end of a sentence token is reached. Figure 5.4a presents the results. We see that in this case, the similarity of all the models with brain representations increases slightly (this could be because of the reduced dimensionality of the similarity matrix), and we see that the general patterns stay similar.

In Figure 5.4b we observe that at the story segments where a name of a character is mentioned, the patterns of similarities change a bit, e.g. the last layer of BERT is less similar to the brain representations compared the first layer of BERT, when an intermediate amount of context is provided to the model. This finding is difficult to interpret, but warrants further research.

Different Regions of the Brain We looked at the similarity scores of the computational representations with the representations at different regions of the brain. This is illustrated in Figure 5.5 for subject 4 as an example. We observe that the patterns of representational similarity of different models are very similar across different brain

regions, i.e. the scores scale for all regions almost similarly across different models. Despite the low correlations between the models and the brain activation, we find that all the models are consistently best aligned with the regions in the Left Anterior Temporal Lobe (LATL). This region is known for semantic and sometimes syntactic processing of language [Bemis and Pylkkänen, 2011, Leffel et al., 2014, Westerlund and Pylkkänen, 2014]. We also find some correlation with the Left Parietal Lobe, which is not known to be responsible for language processing.

It is noteworthy that when considering the average representational similarity score between different brain regions for the eight subjects, both within and across subjects, we find that the different regions of a single brain are more similar ($RSS = 0.4$) than the same regions of different brains ($RSS = 0.12$).

If brain functions involved in story comprehension are spatially localized and brains are organized similarly across individuals, we would expect the same regions from different subjects to be more similar than different regions from the same subject. These are counter-intuitive findings that warrant further investigation.

5.4 Probing for Linguistic Information

Have we really made progress, if all we have done is show a correlation between activity in two distinct systems, both of which remain black boxes? Ultimately, we would like to show that the representa-

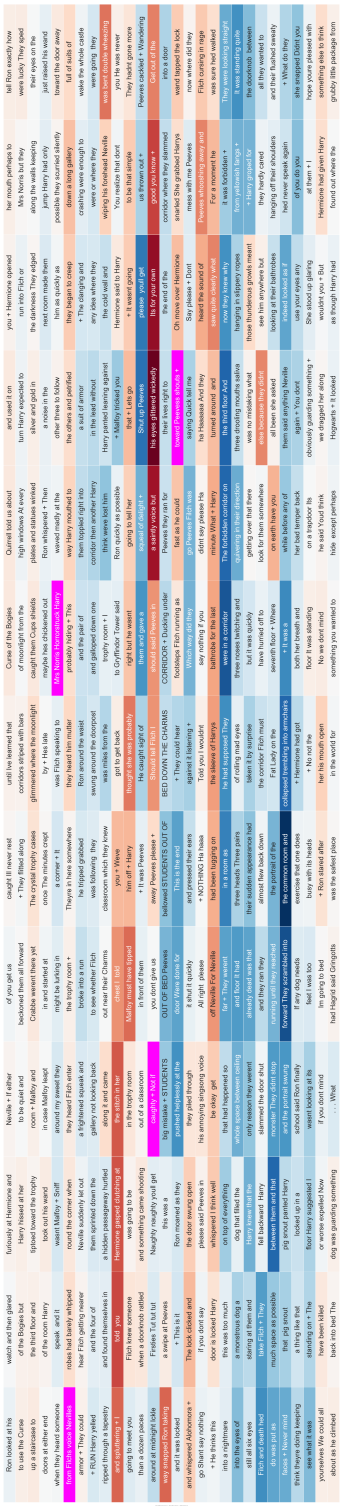


Figure 5.6: Difference in the predictive power of GoogleLM and GloVe-cumulative(weighted) for brain activity patterns at each step of the story for one of the test blocks (Blue: GoogleLM is better, Red: GloVe is better).

tions from language models best predict
brain activations *because* they encode specific linguistic information. However, proving such a causal link is difficult, and in this section, we settle for demonstrating that these representations encode relevant linguistic information (and that this information is more easily extracted from the language model representations than from the brain vectors directly).

A hint that there is a causal link between the linguistic information and the predictive success, can be found in Figure 5.6, which shows the differences in predictive accuracy between the GoogleLM representations and GloVe. A striking observation here is that the GoogleLM representations are relatively most successful in the passages from Harry Potter that contain much dialogue, and hence much quoted speech. This observation motivates the choice for the first linguistic feature, denoted by *InQuote*: (1) whether or not the current word is part of a quotation or not. The other linguistic features that we study (motivated by computational convenience) are: (2) whether the current word is the start of a new sentence (denoted by $< S >$); (3) whether the current word is the end of a sentence (denoted by $< /S >$); (4) The part of speech tag of the current word (denoted by PoS); (5) For pronouns: which character in the story the pronoun is referring to (denoted by *Refs*). Note that the first three sets of features, start/end of a sentence and InQuote, are binary features. For these features, we report the accuracy of the classifier only for the cases when the feature is present (Recall).

To explore the extent to which we can more directly relate brain vectors to specific linguistic information, we tried different approaches such as training diagnostic classifiers [Hupkes et al., 2018] to predict different linguistic features of the story using, or other methods for measuring the correlation or mutual information between different feature vectors and the brain vectors.

For training diagnostic classifiers using the brain vectors, due to the possible delays, we consider brain vectors from previous blocks of words. Thus, we train separate classifiers using not only the labels for the current words, but also labels for the words presented during brain scans at time $t-2$, $t-4$, $t-6$, $t-8$, and $t-10$ (Figure 5.7 and 5.8).

The accuracy of predicting whether a word occurs within a dialogue, i.e., if the word is inside quotes, using the brain vectors is below chance (50%). See all the cases in Figure 5.7. In this figure, we see how the accuracy for predicting the InQuote feature changes for different steps and per each word in the 4-word sequence associated with the brain vector. We see that for the 3$^{\text{rd}}$ and 4$^{\text{th}}$ word, the accuracy rises to its peak at 6s delay, whereas for the first and 2$^{\text{nd}}$ word the peak of the accuracy is at 2s delay.

Looking at Figure 5.8, we see that the 4$^{\text{th}}$ word, from the 4-word ordered sequence associated with the brain vector, gets the highest Mean Reciprocal Rank (MRR, the harmonic mean of the rank of the correct label) score for predicting which character from the story a pronoun refers to. We also notice the jump in its MRR score when we go from delay=0 to delay=2. From then, the MRR score stays the same.

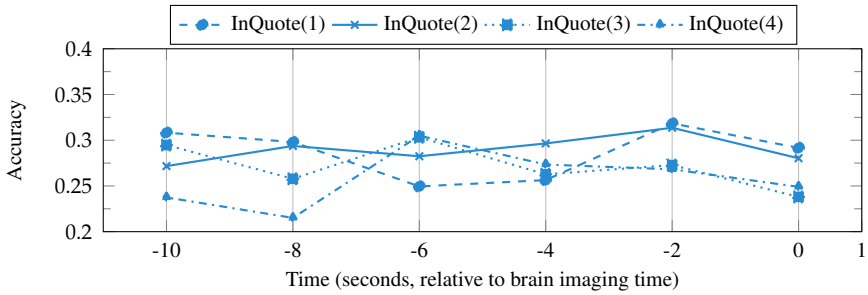


Figure 5.7: Accuracy of predicting different story features from brain vectors. InQoute(i) indicates whether if the *i*th word from the 4 word sequence stimuli is part of a quotation or not. Results obtained for all cases is below chance (50%).

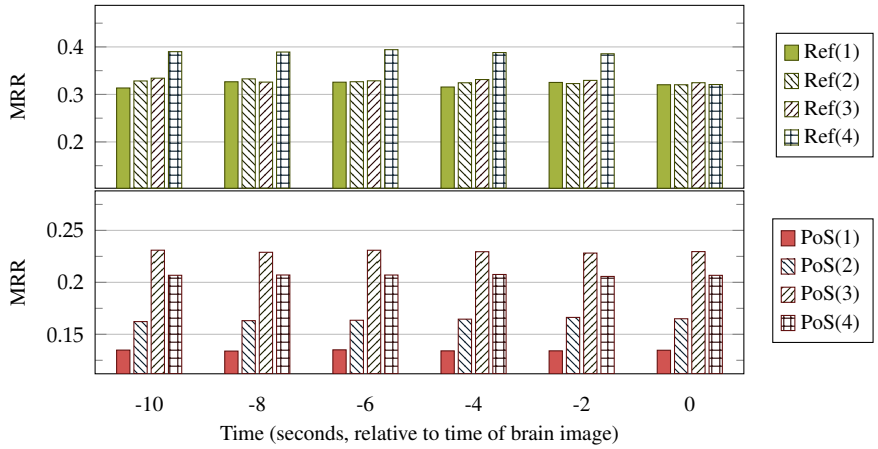


Figure 5.8: MRR (Mean Reciporal Rank) of predicting different story features from brain vectors. Ref(i) indicates which character the *i*th word in the 4 word sequence stimuli refers to and PoS(i) refers to the PoS tag of the *i*th word.

For the PoS feature, the 3rd word in the sequence gets the highest score, but for all the words the MRR score is almost the same for all amounts of delay.

For the features of the start/end of the sentence, we get very low (almost zero) accuracy with classifiers that use brain vectors as input (so we filtered out them from the plots).

Generally, with direct probing, we found no indication that brain signals contain significant information about the story features we looked into (Figures 5.7 and 5.8). Of course, this does not show that this information is not captured by the human brain; it only suggests this information is not accessible for a *linear* classifier, using fMRI data with the given temporal and spatial resolution.

For training diagnostic classifiers using the LSTM internal states, we have tried four cases: using memory/hidden state of the first/second layer of the LSTM for the given time step (Figure 5.9).

As Figure 5.9 shows, the diagnostic classifiers for detecting the start/end of sentences achieve acceptable performance when using GoogleLM internal states (especially the hidden states). These results are much higher compared to the classifiers using the

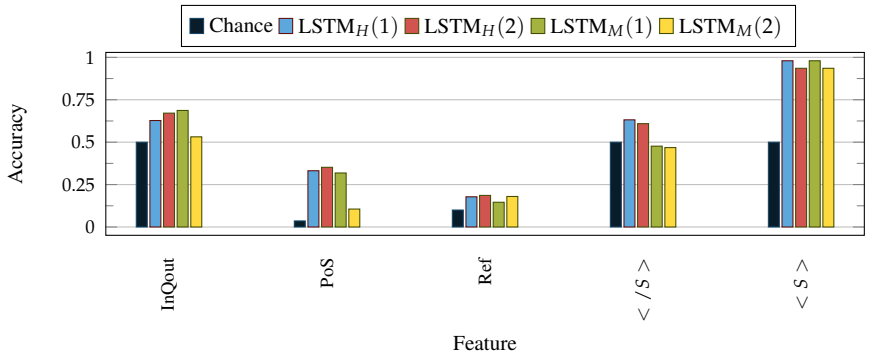


Figure 5.9: Accuracy of predicting different features of the text from the latent variables of the GoogleLM language model. LSTM$_H(i)$ indicates the hidden state of the ith layer of the LSTM language model. LSTM$_M(i)$ indicates the memory state of the ith layer of the LSTM language model.

Brain vectors for the start/end of the sentence. We also see that the PoS as a syntactic feature is well captured by the GoogleLM internal state and the accuracy for predicting this feature is significantly above chance. Note that here we have 28 PoS categories, thus, the chance level is about 0.036. The other two features are the Refs and InQuote features. Both of these features are predictable only slightly above chance from the internal states of the GoogleLM.

The main message from these probing experiments could be to show the deficiency of probing techniques when dealing with high-dimensional brain data, and to highlight the importance of using more complex language models to explain neurological signals and gain a better understanding of how language is implemented and processed in the human brain. But more importantly, these experiments could indicate that neither the representations obtained from these language models nor the brain signals contain the relevant linguistic signals we are searching for.

5.5 Conclusion

In this chapter, we employed a representational similarity metric to compare the representations from the language encoding models and the brain activity patterns, i.e. measure the alignment between the brain activation patterns and activations of the internal state of the models. The main advantage of representational similarity analysis is that it treats both the brain and the model as a blackbox; it does not need to know how brains or models represent objects, words or sentences, but only how similar representations are to each other. For N stimuli considered, the analysis only compares $\frac{1}{2}N(N-1)$ pairs of pairwise similarities (assuming similarities are symmetric), regardless of the dimensionality of two representational spaces. This bottleneck brings many advantages including computational efficiency, reuse of the similarity matrices in multiple comparisons, and not having to worry about how to map representations of very different natures to each other. It also brings important limitations and inevitable

information loss, e.g. standard RSA, assumes all features of the representational spaces to have equal contributions.

It is noteworthy that, in our experiments, we observe more similarities between representations learned by some architectures and brain representations. However, caution is required when interpreting these results, as the representational similarity between all models and the brain images remains very low. Further analysis of various (bigger) datasets is needed to get a better interpretation of what is happening in both the brain and these computational models.

Using brain data to evaluate the representations learned at different layers of each of the language encoding models, we find that layers of the LSTM-based models achieve higher similarity scores with brain data compared to single-word representation models like GloVe and the Transformer based models. This observation could show that the learning biases of the LSTM-based language models are closer to what happens in the human brain. Zooming into the results, we see that while changing the conditions of the inputs to the models has a significant impact on the representations they compute and their performance on NLP tasks (empirical results provided in chapter 2), these changes do not get reflected in their alignment with the brain representations.

Surprisingly, we find that the correlation between the brain signals and different layers of the neural language models we study is higher at the first layer. In chapter 2, we have shown that the context-dependence in these layers is higher in the upper layers (i.e., the second layer in LSTM language models). These two observations alongside each other, make us wonder if what is captured in the fMRI signals about language processing that is reflected in the correlation with representations obtained from language models is merely about low-level linguistic signals and can not tell us much about higher-level processes involved in sentence processing and story comprehension.

Finally, evaluating computational models of language processing with brain imaging data for a task such as "story reading" is hard, because of the inherent issues in the brain data and also the complexity of the task [Beinborn et al., 2019]. Various techniques for representational similarity analysis and the regression approach make it possible to make a bridge between these black boxes, neural network models for language processing on the one hand and the human brain on the other. While each of these approaches has its benefits and limitations, they might provide us with complementary information. Hence, it is important to look at both.

PART

III

Effects of Inductive Biases

Inductive biases are the characteristics of learning algorithms that influence their generalization behavior, independent of training data. They are one of the main driving forces to push learning algorithms toward particular solutions [Mitchell, 1980]. Without proper inductive biases, robust out-of-distribution generalization is not achievable. In the absence of strong inductive biases, a model can be equally attracted to several local minima on the loss surface; and the converged solution can be arbitrarily affected by random variations like the initial state or the order of training examples [D'Amour et al., 2020, Dodge et al., 2020, McCoy et al., 2020, Sutskever et al., 2013].

Inductive biases of neural network models can be rooted in a variety of sources, including architectural constraints, training regimes, optimization algorithms, objective functions, regularization techniques, etc. In this chapter, we use a wide set of analysis tool-kits and metrics, to study the effects of inductive biases. We look into different metrics indicating generalization and out-of-distribution generalization behaviour of the models, as well as illustrating the variation in the solutions they converge to.

In chapter 6, we study one of the most interesting architectures for language models, recurrent neural networks, and investigate the inductive bias of this architecture and empirically, identify the main factors that contribute to the recurrent inductive bias. We show that recurrence, in its different forms, indeed facilitates capturing hierarchical structures in sequential data.

Next, in chapter 7, we aim to study the expressivity of the models to learn a generalizable solution versus the learnability of the generalizable solution for them. The main question here is how can we distinguish between the case where a model is not expressive enough to be able to execute the desired solution versus when the challenge is the learning process and in the lack of proper inductive biases it is difficult for the model to converge to a particular desired solution. We investigate this through the lens of knowledge distillation. The idea is that the teacher model can help in improving the learnability of the desired solution for the student model.

Through an extensive set of experiments we, empirically, study how inductive biases of models are reflected in their output and how in student-teacher setups the inductive biases of the teacher model can impact the solution the student model converges to.

We show that in certain knowledge distillation settings, where we have a teacher with a strong inductive bias providing the supervision signal, the effects of the inductive bias of the teacher transfer to the student model. To put it more clearly, we can see the effects of the inductive biases of the teacher model in the solution the student model converges to. I.e., the student model shows a generalization behaviour similar to its teacher even when the training data used to transfer the knowledge is under-specified.

6
Recurrence

While Transformers do extremely well on many tasks given enough training data and computation [Brown et al., 2020, Devlin et al., 2019, Radford et al., 2019, Vaswani et al., 2017], several studies have shown that LSTMs, the most popular variants of RNNs, can perform better than Transformers on tasks requiring sensitivity to hierarchical (linguistic) structure, especially when the data is limited [Dehghani et al., 2019, Hahn, 2020, Tran et al., 2018]. Theoretically, both RNNs and Transformers can deal with finite hierarchical structures. But, they have different preference inductive biases and the superior performance of LSTMs over Transformers in these cases is attributed to their recurrent inductive bias. The recurrent inductive bias of LSTMs seems to have an important role in enabling them to model the hierarchical structure of the inputs. However, it is not clear exactly what we mean by"recurrent inductive bias". In this chapter, we try to identify the sources of recurrent inductive bias and investigate its impact on the final solution of the models.

This chapter is based on the following paper.

- Samira Abnar, Mostafa Dehghani, and Willem Zuidema. 2020. Transferring inductive biases through knowledge distillation. arXiv preprint arXiv:2006.00555 (2020).
- List of contributions is as follows. Samira Abnar: Preparing the framework for running the experiments, Designing and Running experiments, and Writing the paper. Mostafa Dehghani: Designing the experiments, Helping with some of the visualizations, Reviewing and revising the paper. Willem Zuidema: Guiding the research, Reviewing and revising the paper.

6.1 On the Importance of Recurrent Inductive Bias

Among sequence modeling architectures, models with recursion are in particular powerful for natural language processing due to their adequacy to model hierarchical structures [Linzen et al., 2016]. The recursion in a model can be implemented in various ways, like in Recurrent Neural Networks [Elman, 1990], Recursive Neural Networks [Le and Zuidema, 2014, Socher et al., 2010] and Universal Transformers [Dehghani et al., 2019, Hao et al., 2019]. While theoretically, both recurrent neural networks (RNNs) and Transformers can deal with finite hierarchical structures, empirical results indicate the superiority of RNNs over Transformers [Dehghani et al., 2019, Hahn, 2020, Tran et al., 2018].

In the literature [Dehghani et al., 2019, Sutskever et al., 2013], the inductive bias of RNNs is referred to as the *recurrent inductive bias*. Here, we distinguish between three main sources of this bias:

1. **Sequentiality**: There is an inherent notion of order in the architecture that forces the model to access the next tokens in the input one by one and process them sequentially.

2. **Memory bottleneck**: The model has no direct access to the past tokens and has to compress all the information from the past in a hidden state, which is accessible when processing a new token.

3. **Recursion**: The model recursively applies the same function on the varying input at every step.

Transformers [Vaswani et al., 2017], in contrast, process the input in parallel. Although a weak notion of order is encoded by positional embeddings, no explicit assumption is made in the connectivity structure of the architecture. Moreover, they have a global receptive field and can access all tokens through self-attention. Finally, standard Transformers are not recursive. However, the standard Transformer can be modified to have an architecture with specifications that are similar to RNNs.

The research question we address in this chapter is: What is the contribution of each of the above-mentioned factors in the recurrent inductive bias of RNNs and can we improve the capability of Transformers in capturing hierarchical structures in input sequences by incorporating each of these factors into the standard Transformer architecture?

Our goal is to empirically demonstrate the benefits of the different components of the recurrent inductive bias. For this purpose, we develop experiments with variants of Transformers in which we attempt to approximate some of the RNNs' assumptions. We find that by adding increasingly more components of recurrence to Transformers, their behavior becomes more similar to LSTMs.

Table 6.1: Performance (mean±std over 4 trials) of different LSTM and Transformer models trained independently with the LM objective.

Model	Perplexity ↓	$\mathcal{D}-$Accuracy ↑	$\mathcal{A}-$Accuracy ↑
Transformer	57.5 ± 0.12	0.942 ± 0.0017	0.919 ± 0.0018
Small Transformer	**55.3** ± 0.08	0.947 ± 0.0012	0.926 ± 0.0020
LSTM	56.7 ± 0.09	**0.951** ± 0.0012	**0.940** ± 0.0024
Small LSTM	58.0 ± 0.11	0.949 ± 0.0006	0.937 ± 0.0015

6.2 Experimental Setup

To examine the effects of different sources of the recurrent inductive bias, we study the performance of LSTMs and variants of Transformers on the task of predicting number agreement between subjects and verbs in English sentences.

6.2.1 Dataset and Tasks

We use the subject-verb agreement dataset of [Linzen et al., 2016], for which the size of the training set is ∼121k examples and the size of the test set is ∼1m. Succeeding at this task is a strong indicator that a model can learn syntactic structures and is therefore proposed by [Linzen et al., 2016] as a proxy for assessing the ability of models to capture hierarchical structure in natural language. It is shown that RNNs have better inductive biases to learn this compared to standard Transformers [Dehghani et al., 2019, Tran et al., 2018]. In this task, examples are grouped into different levels of difficulty based on the number of "agreement attractors"[1], and distance (number of all words) between the verb and its subject.

Task setup Similar to [Tran et al., 2018], we follow two setups: 1) when the learning objective is next word prediction, i.e., language modeling (LM); 2) when we directly optimize for predicting the verb number, singular or plural, i.e., classification. In the LM setup, we look at the probabilities predicted when the target of the prediction is the verb of interest, and see whether the probability of the correct form of the verb is higher than the other form (singular vs plural). In the classification setup, the input to the model is a sentence up to the position of the verb of interest and the model predicts whether the verb at that position is singular or plural.

6.2.2 Model Architectures

In the LM setup, we employ two unidirectional LSTMs with different sizes, *LSTM* and *Small LSTM*, and two Transformers, *Transformer* and *Small Transformer*. In this setup, corresponding LSTMs and Transformers have roughly the same number of

[1] Agreement attractors are intervening nouns with a different number than the number of the subject. E.g., given the input "`The **keys** to the cabinet (is?/are?).`", the word "cabinet" is an agreement attractor.

Table 6.2: Performance (mean±std over 4 trials) of different LSTM and Transformer models trained independently with the classification objective.

Model	$\mu-$**Accuracy** ↑	$\mathcal{D}-$**Accuracy** ↑	$\mathcal{A}-$**Accuracy** ↑
Transformer	0.954 ± 0.0016	0.901 ± 0.0037	0.717 ± 0.0244
Transformer-seq	0.964 ± 0.0010	0.909 ± 0.0037	0.742 ± 0.0121
UniversalTransformer-seq	0.969 ± 0.0004	0.932 ± 0.0055	0.806 ± 0.0153
LSTM	**0.977** ± 0.0001	**0.970** ± 0.0003	**0.928** ± 0.0007

parameters. In the classification setup, we compare the following models: (1) a standard unidirectional LSTM (*sequentiality* + *memory bottleneck* + *recursion*) (2) Transformer: Transformer encoder with a class token (`CLS`) for classification, BERT [Devlin et al., 2019] style, (3) Transformer-seq: Transformer encoder with future masking where the classification is done using the representation of the last token[2] (*sequentiality*), (4) UniversalTransformer-seq: Universal Transformer [Dehghani et al., 2019] encoder, in which the parameters are shared in depth, with future masking (*sequentiality* + *recursion*).

6.2.3 Evaluation Metrics

To compare the performance of the models on the subject-verb agreement task, we report macro accuracy over different groups of examples in the test set in terms of distance ($\mathcal{D}-$Accuracy) and numbers of attractors ($\mathcal{A}-$Accuracy). Since the number of examples that fall into different categories based on these two factors is not balanced, evaluating the accuracy of each category independently and reporting the macro average helps us capture the difference between the ability of the models to solve examples with different levels of difficulty.

As an indicator of the general performance of the models, in the classification setup, we report micro accuracy ($\mu-$Accuracy) and in the language modelling setup, we report perplexity. Additionally, in the classification setup, besides the accuracy, we evaluate how calibrated the models are in terms of the Expected Calibration Error (ECE). This metric reflects how the confidence of the model is correlated with its correctness. Theoretically, a model can have high accuracy but have high calibration error, hence we treat this as another indicator of performance that is not necessarily correlated with accuracy. We believe for models with similar accuracy, having a lower calibration error can be an indicator of better generalization.

[2]Note that future tokens are masked out by default when using a transformer in the decoder mode, e.g., in LM setup.

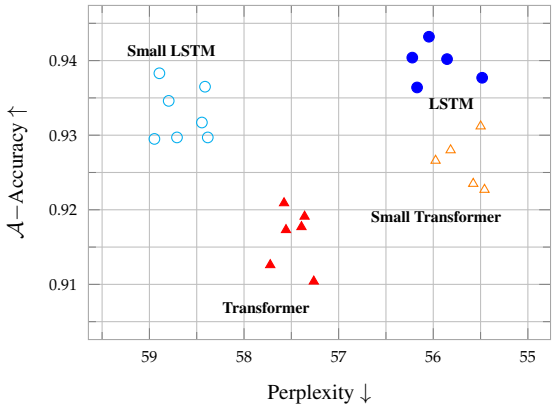


Figure 6.1: $\mathcal{A}-$Accuracy vs perplexity (high to low from left to right) for language models of different architectures and sizes.

6.3 Examining the Impact of Recurrent Inductive Bias

In this section, we report results that illustrate the merits of the recurrent inductive bias. Table 6.1 shows the performance of the models when trained with the LM objective. A first important observation, in line with the results of [Tran et al., 2018], is that LSTMs achieve better accuracy on the subject-verb agreement task compared to Transformers. Even for instances of Transformer language models that achieve better (lower) perplexity, the accuracy on this task is worse compared to LSTM instances.

Since both models achieve good scores on the training set, this suggests that LSTMs better capture relevant patterns, such as the hierarchical structure of the input, which leads to better generalization on this task.

Figure 6.1 illustrates the accuracy versus perplexity of several instances of each model, in the LM setup. Note that although perplexity is an indicator of how well the model is optimized given the objective function, the accuracy is the metric that matters and shows models' generalization in the subject-verb agreement task (In chapter 7, we show how using KD the behavior of Transformers changes in terms of accuracy versus perplexity and become more similar to LSTM teachers).

There is another interesting observation in Figure 6.1. In this plot, for each model, we have two different settings: large and small variants, as measured by the number of trainable parameters. More parameters for a model, given a fixed architecture, means richer hypothesis spaces. We can see that while for the LSTMs, increasing the size of the model results in better performance, for the Transformers increasing the number of parameters results in a worse performance. This aligns with the bias-variance trade-off argument that when using a model with weaker biases for the task at hand, in this case, Transformers, if we fix the amount of data, richer hypothesis spaces may hurt the generalization because they increase variance. In contrast, adding more capacity leads to better accuracy in LSTMs as their stronger inductive biases control the generalization error.

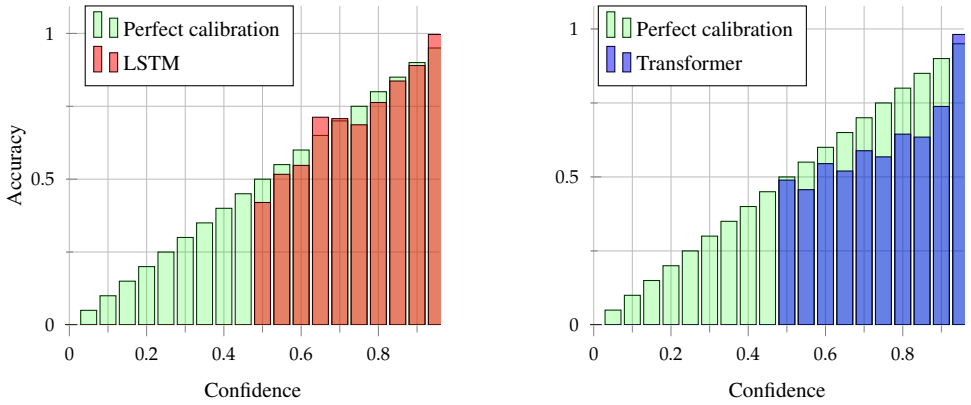
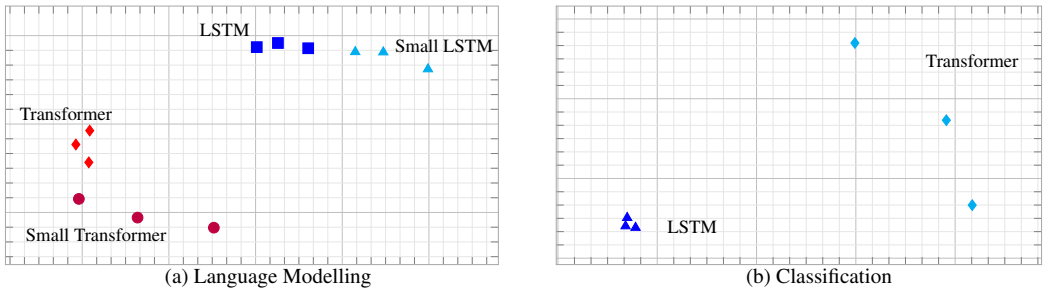


Figure 6.2: Calibration plots for an LSTM and a Transformer model trained on subject-verb-agreement classification. Note that since the task is binary classification, accuracy for confidences lower than 0.5 is not defined.



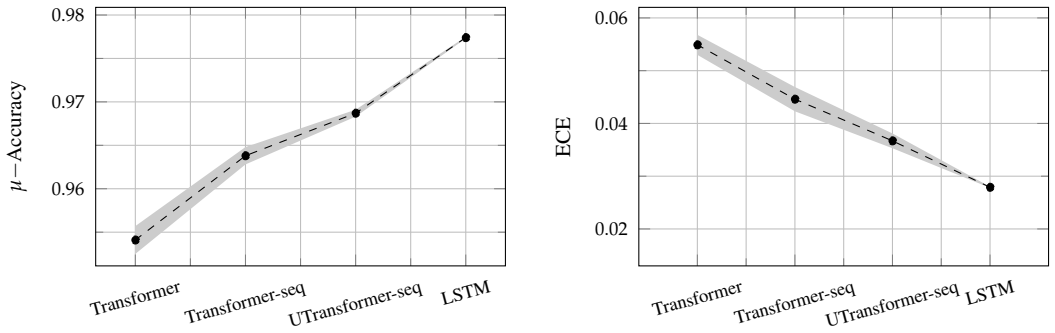
(a) Language Modelling (b) Classification

Figure 6.3: 2D projection of representational similarity of the activations from the penultimate layers for 1000 examples from the validation set (check Appendix 7.3.2 for more details). We use the notation of $a \rightarrow b$ to refer to the student model b distilled from teacher model a.

6.4 Examining the Sources of Recurrent Inductive Bias

In Table 6.2 we show the results of models trained on the classification objective. We compare LSTM with variants of Transformers with different inductive biases. The table shows that similar to the LM results, LSTM achieves the best performance. Interestingly, comparing all four models, we find that the performance steadily increases as more aspects of the recurrent inductive bias are included. This is illustrated in Figure 6.4a, with the filled circles on the black, dashed line.

As another indicator of the quality of the solutions that different models converged to in the classification setup, we look into their confidence calibration. Confidence calibration captures how well the likelihood (confidence) of the prediction of the model predicts its accuracy [Guo et al., 2017]. For a well-calibrated model, if we bin the confidence scores and compute the accuracy for each bin, the accuracies are perfectly correlated with the confidence values. The Expected Calibration Error (ECE) is computed as the distance between the calibration curve of the model and the perfect calibration curve [DeGroot and Fienberg, 1983]. In Figure 6.4b, we plot the ECE [Guo et al., 2017] of the models in the classification setup, with the filled circles on the black dashed line. In



(a) Accuracy ↑ (b) Expected Calibration Error ↓

Figure 6.4: Performance (mean±std over 4 trials) of models with different inductive biases trained independently or using KD with different teachers.

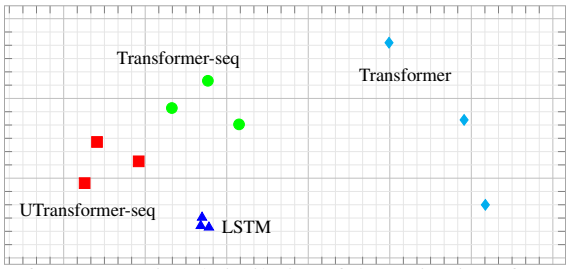


Figure 6.5: 2D projection of representational similarity of the activations from the penultimate layers for 1000 examples from the validation set.

line with the trends in the performances of these models, the expected calibration error decreases as we move from Transformer toward LSTM.

Additionally, as shown in Figures 6.4a and 6.4b, we find a decreasing trend in the variance of the models, i.e., adding more inductive biases to the models decreases their variance. This is empirical evidence that supports the relation between the variance of the solutions a model converges to and its inductive biases.

6.5 Conclusion

Several studies have shown that the recurrent inductive bias of LSTMs, the most popular variants of RNNs, helps these models deal with tasks requiring sensitivity to hierarchical (linguistic) structures. In particular, when data and compute are limited, LSTMs perform better than models such as Transformers variants that lack the recurrent inductive bias (Tran et al., 2018; Dehghani et al., 2019).

In this chapter, We provide empirical evidence to support the importance of recurrence for solving the subject-verb-agreement task [Dehghani et al., 2019, Tran et al., 2018], as an example of tasks that require capturing hierarchical structure in the input sequence.

We show that LSTMs not only achieve a higher accuracy, but also better confidence

calibration. We also show that LSTMs and Transformers converge to solutions with very different representational spaces. Additionally, we show that when these models are not directly trained for this task, but just on a language modelling objective (a transfer learning setup), even with higher (worse) perplexity, LSTMs achieve higher accuracy on the subject-verb agreement task. These experiments and results confirm that LSTMs have inductive biases that standard Transformers lack and this inductive bias helps them to better capture hierarchical structures in the data.

Furthermore, we identify and examine different sources of the recurrent inductive bias. We distinguish between the three main sources of the recurrent inductive bias: Sequentially, Memory bottleneck and Recursion and by applying these constraints on the basic Transformer architecture we illustrate that each of these can significantly impact the solution of a model.

7

Transferring the Effects of Inductive Biases

Different inductive biases of learning algorithms can drive them towards solutions with inherently different characteristics even when they achieve similar performance with respect to the training objective. Our focus in this chapter is to understand whether the effects of inductive biases rooted in the architecture of a neural network can transfer through knowledge distillation to another model depending on its inductive biases and expressivity. The answer to these questions helps us to understand the limits of knowledge distillation, and its potential as a comparison framework to study inductive biases and the expressive power of different models. Our experiments indicate that training models through KD from a teacher with a strongly different inductive bias significantly affects the characteristics of the solutions the student models converge to. This effect is not necessarily persistent and in the absence of evidence for the desirability of a solution in the training data. The student model could eventually forget those effects if we stop distillation and continue the training with ground-truth labels. Nevertheless, initializing a model through KD from a teacher model with different inductive biases

---

This chapter is partially based on the following paper (some of the contents are not published before).

- Samira Abnar, Mostafa Dehghani, and Willem Zuidema. 2020. Transferring inductive biases through knowledge distillation. arXiv preprint arXiv:2006.00555 (2020).
- List of contributions is as follows. Samira Abnar: Preparing the framework for running the experiments, Designing and Running experiments, and Writing the paper. Mostafa Dehghani: Designing the experiments, Helping with some of the visualizations, Reviewing and revising the paper. Willem Zuidema: Guiding the research, Reviewing and revising the paper.

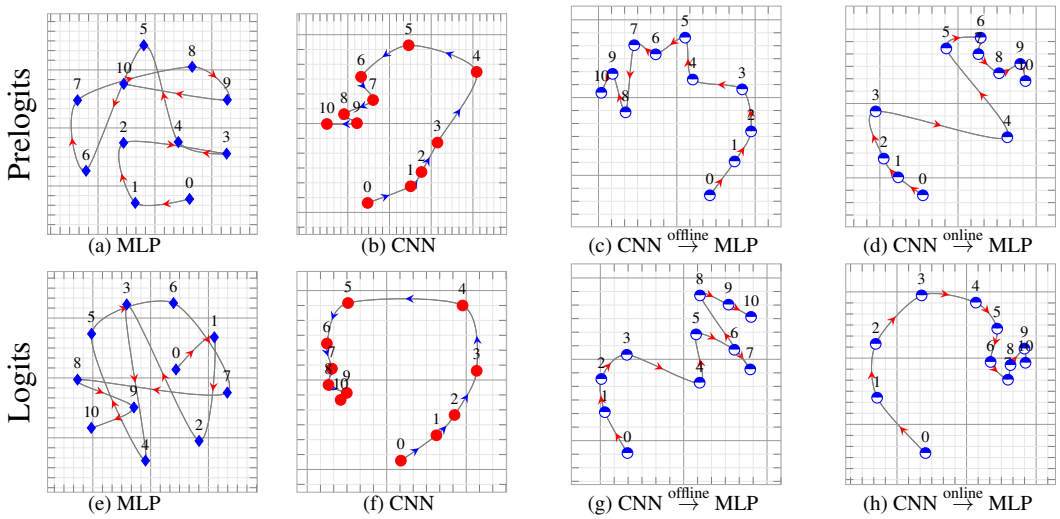


Figure 7.1: Training path of models in terms of relational changes in the representational space of the models. In these plots, each point represent the state of the model at a specific epoch, from the initial state to the convergence. The visualization is based on a 2D projection of the representational similarity of the activations from the penultimate and logits layer for 1000 examples from the validation set, i.e. Translated MNIST (more details on Appendix 7.3.2).

could be beneficial in some cases.

7.1 Knowledge Distillation and Biases

KD refers to the process of transferring knowledge from a teacher model to a student model, where the logits from the teacher are used to train the student. KD is best known as an effective method for model compression [Buciluǎ et al., 2006, Hinton et al., 2015, Sanh et al., 2019] which allows taking advantage of a huge number of parameters during training while having an efficient smaller model during inference.

The advantage of KD goes beyond model compression. For example, self-distillation, distilling a model into itself iteratively, can sometimes lead to improvements in performance [Furlanello et al., 2018, Mobahi et al., 2020]. KD can also be used to combine strengths of different learning algorithms [Geras et al., 2016, Kuncoro et al., 2019, 2020, Touvron et al., 2020]. Different algorithms vary in terms of computational/memory efficiency at training/inference or inductive biases for learning particular patterns. This makes them better at solving certain problems and worse at others, i.e., there is no "one size fits all" learning algorithm, and KD can be exploited to find better trade-offs.

Considering the popularity and the different use cases of KD, it is important to explore the potentials and limits of KD to understand how and to what extent a model is affected when trained through KD in contrast to when it is trained with ground-truth labels beyond the boost in its performance and speed up in training time. Understanding this is extremely important when we are interested in the downstream performance or

out-of-distribution behaviour of the models. E.g., consider a scenario where we apply KD to transfer the knowledge of one model to another using the upstream objective and training data. Can we expect the student model to have the same behaviour on the downstream tasks as the large model, merely based on the fact that they achieve the same level of performance on the upstream task? While, even for models with similar architecture and size, different trained instances can have very different behaviours on downstream tasks, which can be explained by under specification of the objective in the training data [D'Amour et al., 2020], KD can pose a strong bias on the student model and derive it toward the solution of the teacher model.

In this chapter, we ask: "*In KD, can the effects of the inductive biases of the teacher on its final solution, which are also reflected in the information encoded in its output logits [Hinton et al., 2015]., be transferred to the student model?*". We are specifically interested in cases where the student model can realize functions that are realizable by the teacher, i.e., the student model is efficient with respect to the teacher model [Cohen et al., 2016], while the teacher has a *preference inductive bias* so that the desired solutions are easily *learnable* for the teacher [Seung et al., 1991].

We consider two scenarios where the teacher and the student are neural networks with heterogeneous architectures, hence have different inductive biases. We train the models, both independently and using KD. We chose tasks in which one of the models is able to achieve a generalization performance more similar to humans than the other models. We refer to the better model as the model with the "right" inductive biases for learning the task. One potential criticism about this idea could be why the outputs of the teacher model could provide a better signal for the student model than the ground-truth signals (which are outputs of humans). Is it not the case that the ground truth signals should reflect the effects of the inductive biases of the human brain? One potential response to this point could be that the role of the teacher model in the KD process is to learn an approximation of the ideal solution which is more learnable for the student model compared to the perfect solution.

In the first test case, we study RNNs vs. Transformers [Vaswani et al., 2017], on the subject-verb agreement prediction task [Linzen et al., 2016]. In this task, we use LSTMs [Hochreiter and Schmidhuber, 1997] as the most widely used RNN variant. LSTMs are shown to perform better than vanilla Transformers in this task and their superior performance is attributed to their so-called "recurrent" inductive bias [Tran et al., 2018]. In chapter 6, we identify the sources of the recurrent inductive bias of LSTMs: *sequentiality*, *memory bottleneck*, and *recursion*, and demonstrate the benefits of each. Here, we show that through distilling knowledge of LSTMs to Transformers, the solutions that the Transformer models learn become more similar to the solution learned by LSTMs.

In the second test case, we study CNNs vs. MLPs, in the context of the MNIST-C (Corrupted MNIST) benchmark [Mu and Gilmer, 2019], which is designed to measure the out-of-distribution robustness of models. We train our models on MNIST and

evaluate them on the Translated/Scaled MNIST. The particular form of parameter sharing in CNNs combined with the pooling mechanism makes them equivariant to these kinds of transformations [Goodfellow et al., 2016], which leads to better generalization in these scenarios compared to MLPs.

In our experiments and analysis on these two test cases[1], we compare the behavior of different models, from a wide range of perspectives, when trained in different setups including **(1)** when trained without KD, but directly from the data, **(2)** when trained with KD using a teacher with a similar architecture to the student, i.e. self-distillation, and **(3)** when trained with KD using a teacher with a different architecture that has stronger inductive biases that suit the task, compared to the student.

As the first step, in setup (1), i.e., no KD, we demonstrate how inductive biases arising from different architectural choices affect the generalization behavior of the models we study. We show that the models with more suitable inductive biases not only have better accuracy but also that the solutions they converge to are better in terms of other metrics. We also show that different instances of the model with stronger inductive biases have less variance in terms of all the metrics.

Then, we apply KD to train the models and contrast the behavior of models trained with the setups (2) and (3) with the models trained with setup (1), i.e. with KD vs. without KD. We show that regardless of the properties of the teacher, KD is a powerful technique in which the teacher model drives the student toward a particular set of solutions that is more restricted compared to the set of possible solutions that a student can converge to when it learns directly from data.

Next, as the main contribution of our experiments in this chapter over previous works that study KD, we contrast the behavior of models trained with setup (3) with the models trained with setups (1) and (2):

- We show the performance of the student models in setup (3) increases, not only on in-distribution test sets, but also on out-of-distribution data. We demonstrate that this happens when the teacher has the right inductive bias and not necessarily otherwise, i.e., setup (2).
- In setup (3), besides performance, we show that, the solution that a student model converges to shares similar characteristics with the solution of its teacher. For instance in terms of confidence calibration, and per sample behaviour of the model (§7.3.3).
- We demonstrate that although the student model is merely exposed to the final logits of the teacher, the structure of the latent space of the student model becomes similar to the teacher, i.e. relational similarity of the internal representations from the student and its teacher increases.

[1] The codes for the input pipelines, models, analysis, and the details of the hyper-parameters used in our experiments are available at *https://github.com/samiraabnar/Reflect*.

As an example, in our second test case (MNIST-C), when training an MLP model with KD using a CNN teacher, the student model explores the solution space in ways more similar to its teacher. Figure 7.1 visualizes and compares the path that an MLP takes during training (Figure 7.1e), compared to a CNN (Figure 7.1f). The CNN model explores the surface in a completely different manner than the MLP, while the path of a student MLP distilled from the CNN model as the teacher (Figure7.1g) is more similar to the CNN.

7.2 Knowledge Distillation in Neural Networks

Knowledge Distillation is a technique that transfers knowledge from one model to another [Hinton et al., 2015]. Hinton et al. [2015] suggest that the power of KD is mostly in being able to transfer the useful information that is embedded in the soft targets of the teacher model, e.g., the relation between the output classes as captured by the teacher model. Hinton et al. [2015] refers to this as the *dark knowledge*. Phuong and Lampert [2019] studies KD from a theoretical point of view in a simplified setting where the task is a binary classification, and teacher and student are linear models. They attribute the success of distillation to three main factors: (1) data geometry, (2) optimization bias, and (3) strong monotonicity. And more recently Tang et al. [2020], conduct extensive analysis and identify three sources for why KD helps: (1) label smoothing, (2) example re-weighting based on teacher's confidence, and (3) prior knowledge of optimal output layer geometry.

The most well-known use of KD is to compress a large, unwieldy model or an ensemble model into a smaller model. Empirically, many people have found that bigger models are easier to train (often explained with the 'lottery ticket hypothesis' [Frankle and Carbin, 2019]); KD makes it possible to distill the knowledge in the large model into a much smaller model, and thus in some sense offer the best of both worlds [Buciluǎ et al., 2006, Hinton et al., 2015, Srinivas and Babu, 2015]. Distilling knowledge from a very big model or an ensemble of models with similar or heterogeneous architectures that are trained on the same or different tasks into a single model with much fewer parameters can lead to similar or sometimes even better performance compared to the teachers [Hinton et al., 2015, Kim and Rush, 2016, Liu et al., 2019a, Luo et al., 2019, Tan et al., 2019].

Previous work has examined the effectiveness of KD in different settings: where the teacher is bigger than the student, but both have similar building blocks [Hinton et al., 2015, Kim and Rush, 2016, Sanh et al., 2019]; where teacher and student are of similar size and architecture [Freitag et al., 2017, Furlanello et al., 2018]; or where the student and teacher have fundamentally different architectures [Ahn et al., 2019, Frosst and Hinton, 2017, Geras et al., 2016, Kuncoro et al., 2019, 2020, Luo et al., 2019, Tang et al., 2019, Touvron et al., 2020].

Table 7.1: Overview of the applications of knowledge distillation in neural networks

Main Purpose	Architectue	Size	Train Data	Test Data	Related Work
Model Compression	Same	$\|\mathcal{M}_{std}\| < \|\mathcal{M}_{tchr}\|$	$\mathcal{S}_{std} = \mathcal{S}_{tchr}$	i.i.d	[Buciluǎ et al., 2006] [Hinton et al., 2015] [Kim and Rush, 2016] [Sanh et al., 2019]
Regularization	Same	$\|\mathcal{M}_{std}\| >= \|\mathcal{M}_{tchr}\|$	$\mathcal{S}_{std} = \mathcal{S}_{tchr}$	i.i.d	[Furlanello et al., 2018] [Freitag et al., 2017] [Mobahi et al., 2020]
Explanation	Different	$\|\mathcal{M}_{std}\| < \|\mathcal{M}_{tchr}\|$	$\mathcal{S}_{std} = \mathcal{S}_{tchr}$	i.i.d	[Craven and Shavlik, 1995] [Craven, 1996] [Frosst and Hinton, 2017]
Combining Pros	Different	$\|\mathcal{M}_{std}\| >=< \|\mathcal{M}_{tchr}\|$	$\mathcal{S}_{tchr} \subseteq \mathcal{S}_{std}$	i.i.d various downstream tasks	[Chan et al., 2015] [Geras et al., 2016] [Luo et al., 2019] [Kuncoro et al., 2019] [Kuncoro et al., 2020] [Touvron et al., 2020]

KD has also been proposed as an interpretation technique, where the knowledge of a big complex model is distilled into a more interpretable model [Craven, 1996, Craven and Shavlik, 1995, Frosst and Hinton, 2017]; Or as a method to compare the capacity and expressiveness of different models [Maheswaranathan et al., 2019, Saxe et al., 2018].

Table 7.1 categorizes and summarizes the many different papers that study the effectiveness of different forms of KD.

Offline Distillation In most cases, KD is applied in an offline setting, i.e., we first train the teacher network and use the trained teacher to train the student, while the parameters of the teacher are fixed. This is the standard distillation process introduced by [Buciluǎ et al., 2006, Hinton et al., 2015]. We apply this setup in our experiments since it is the most common approach. There are other possible settings for KD, e.g. online distillation, where teacher and student models are trained simultaneously.

Online Distillation In online distillation, the teacher and student networks are trained at the same time. The parameters of the student model can get updated with the same mini-batch as the teacher or a different mini-batch. In either case, similar to offline KD, the targets for the student model are the output activations of the teacher. E.g., Anil et al. [2018] uses an online distillation setup, where models are trained on different parts of the dataset and share their knowledge during training. One main advantage of online distillation could be benefiting from the implicit information in the trajectory the teacher model follows before converging to a solution.

Distillation Loss There are several different ways of computing the distillation loss: using only the output of the teacher or taking intermediate layers into account as well [Ahn et al., 2019, Anil et al., 2018, Buciluǎ et al., 2006, Hinton et al., 2015, Park et al., 2019, Sun et al., 2019b, Tung and Mori, 2019]. Potentially, using these alternative losses could lead to transferring different kinds of knowledge depending on the tasks and the configurations of the models. While it is worth doing a thorough comparison of all these techniques, in this chapter we have focused on the most commonly used loss introduced by [Hinton et al., 2015], which is based on the Kullback-Leibler divergence

between output distributions of the teacher, i.e., soft targets, and the output distributions of the student. The output distributions of the teacher and student model, P_t and P_s, are computed similarly, with Equation 7.1.

$$\frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}, \tag{7.1}$$

where $\tau > 1$ is the softmax temperature and z is the logits from the model.

The distillation loss is: $\mathcal{H}(P_t, P_s)$, where $\mathcal{H}$ is the cross entropy loss and is computed as:

$$\mathcal{H}(P_t, P_s) = -\sum_x P_t(x) \log P_s(x) \tag{7.2}$$

When KD is applied as a means for model compression, it is common to compute the total loss as a mixture of distillation loss and actual loss. Since, our focus in this paper is on how much the student model can learn from the teacher model, in our experiments we use pure distillation.

7.3 Distilling LSTMs into Transformers

LSTMs and Transformers are the basic building blocks of many state-of-the-art models for sequence modeling and natural language processing. Transformers are an expressive class of models that do extremely well on many tasks where the training data is adequate in quantity [Brown et al., 2020, Devlin et al., 2019, Keskar et al., 2019, Radford et al., 2019]. Several studies, however, have shown that LSTMs can perform better than Transformers on tasks requiring sensitivity to (linguistic) structure, especially when the data is limited [Dehghani et al., 2019, Tran et al., 2018].

We chose the subject-verb agreement prediction task, introduced by [Linzen et al., 2016], as the test case, as it yields a meaningful difference between LSTMs and Transformers [Tran et al., 2018]. We compare these two families of models and conduct experiments to emphasize the differences between them when trained independently and through KD.

7.3.1 Models Architectures and Training setup

For the subject-verb agreement task, we study Transformers and LSTMs. In the LM setup, we use two sizes for each architecture: LSTM: two-layer uni-direction LSTM, with a hidden size of 1024. Small LSTM: two-layer uni-direction LSTM, with a hidden size of 512. Transformer: six-layer Transformer decoder with a hidden size of 512 and 8 heads. Small Transformer: Transformer: six-layer Transformer decoder with a hidden size of 256 and 8 heads.

Table 7.2: Performance (mean±std over 4 trials) of different LSTM and Transformer models with LM objective when we apply pure distillation with $\tau = 1$.

Student Model		Teacher Model			
		LSTM	**Small LSTM**	**Transformer**	**Small Transformer**
LSTM	$\mathcal{A}-$Accuracy ↑	0.928 ± 0.002	0.931 ± 0.001	**0.908** ± 0.004	**0.926** ± 0.003
	perplexity ↓	59.45 ± 0.019	60.92 ± 0.019	60.01 ± 0.033	58.65 ± 0.004
Small LSTM	$\mathcal{A}-$Accuracy ↑	0.922 ± 0.002	0.927 ± 0.003	0.899 ± 0.006	0.916 ± 0.002
	perplexity ↓	62.52 ± 0.107	63.44 ± 0.027	63.45 ± 0.0644	61.62 ± 0.062
Transformer	$\mathcal{A}-$Accuracy ↑	**0.930** ± 0.003	**0.932** ± 0.001	0.896 ± 0.002	0.920 ± 0.002
	perplexity ↓	**57.03** ± 0.009	**59.09** ± 0.013	**57.67** ± 0.009	**56.64** ± 0.035
Small Transformer	$\mathcal{A}-$Accuracy ↑	0.920 ± 0.002	0.923 ± 0.001	0.883 ± 0.003	0.91 ± 0.001
	perplexity ↓	57.84± 0.027	59.73 ± 0.017	58.44 ± 0.035	57.16 ± 0.009

Table 7.3: $\mu-$Accuracy ↑ (mean±std over 4 trials) of different LSTM and Transformer models with classification objective when we apply pure distillation with $\tau = 1$.

Student Model	Teacher Model			
	Transformer	**Transformer-seq**	**UTransformer-seq**	**LSTM**
Transformer	0.956 ± 0.0013	0.956 ± 0.0006	0.957 ± 0.0027	0.960 ± 0.0008
Transformer-seq	0.960 ± 0.0006	0.963 ± 0.0008	0.968 ± 0.0005	0.972 ± 0.0017
UTransformer-seq	0.961 ± 0.0006	0.964 ± 0.0004	0.969 ± 0.0008	0.975 ± 0.0003
LSTM	**0.968** ± 0.0002	**0.969** ± 0.0011	**0.974** ± 0.0004	**0.976** ± 0.0001

In the classification setup, we employ an LSTM and three variants of Transformer, where the LSTM has a two-layer with a hidden size of 256, and the Transformers have 6 layers, 8 heads, and a hidden size of 128. We use a hidden size of 256 for the UniversalTransformer-seq since its parameters are shared in depth and with the same hidden size as other Transformers, it will have fewer parameters.

For training the independent models we use the Adam optimizer [Kingma and Ba, 2014] with exponential decay learning rate scheduler and for the student models in the distillation process, we use Adam optimizer with cosine decay restart [Loshchilov and Hutter, 2017] learning rate scheduler. The hyperparameters related to the regularization and learning rate schedulers are tuned separately for each model/experiment. For each model, we report the set of hyper-parameters that gives the best average performance across multiple trials with different random seeds for initialization.

7.3.2 Transferring the Effect of Recurrent Inductive Bias

In this section, we show that distilling knowledge from LSTM to Transformer can close the gap between their performance by pushing the Transformer to converge to solutions more similar to LSTM's.

Table 7.2 and Table 7.3 summarize the distillation results, when the training objective is language modeling and classification respectively. A first general observation is that, for these tasks and setups, distilling a model into an identical model could result in a decrease in performance. Note that whether self-distillation results in improved

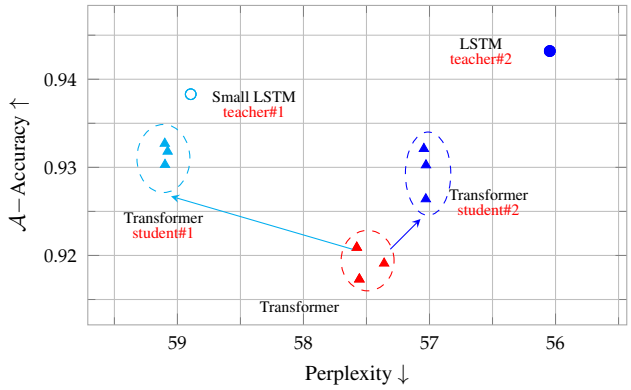


Figure 7.2: $\mathcal{A}-$Accuracy ↑ vs perplexity ↓ (high to low from left to right) for student Transformer with LM objective. In this figure the triangle marks indicate Transformer models, and circles indicate the LSTM teachers used to train the student Transformers.

performance could potentially depend on many different factors such as the architecture of the model, optimization algorithm, and details of the distillation process [Furlanello et al., 2018, Mobahi et al., 2020].

Despite no significant changes in the performance with self-distillation, we can improve the performance of the Transformers through distillation from LSTM teachers.

To check whether this improvement is due to the transfer of the effect of inductive biases through distillation and whether distillation helps students to converge to solutions similar to their teachers, we run a series of analyses. In Figure 7.2 we see how teacher LSTMs pull student Transformers toward solutions with higher accuracy on the subject-verb agreement task in the LM setup. This happens even when the perplexity of the student Transformer is higher (worse) than the independent Transformer.

Figure 6.4, also shows the effects of distillation on each of the four models we study in the classification setup. In Transformer-based models, we get the most significant improvement both in accuracy and ECE when the teacher is an LSTM. As the recurrent inductive biases of the teacher get weaker, the amount of improvement in the performance of student models decreases. Figure 7.3 shows the effect of KD on the calibration, given a student Transformer and an LSTM teacher.

Is the improvement in calibration merely the product of using soft targets? Mueller et al. [2019] shows training neural networks with soft targets (e.g. through label smoothing) results in models that are better calibrated. On the other hand, KD has a regularization effect similar to label smoothing [Tang et al., 2020, Yuan et al., 2019]. Given the lack of significant improvement in ECE in the self-distillation experiments (Figure 6.4b), it is more likely that the cause of the improvement in ECE when distilling LSTMs into Transformers is beyond the label smoothing effect of KD.

To further explore and better understand the effects of KD, we compare the internal representations of these models besides their final output.

Visualisation of representational similarity of the activations from the penultimate

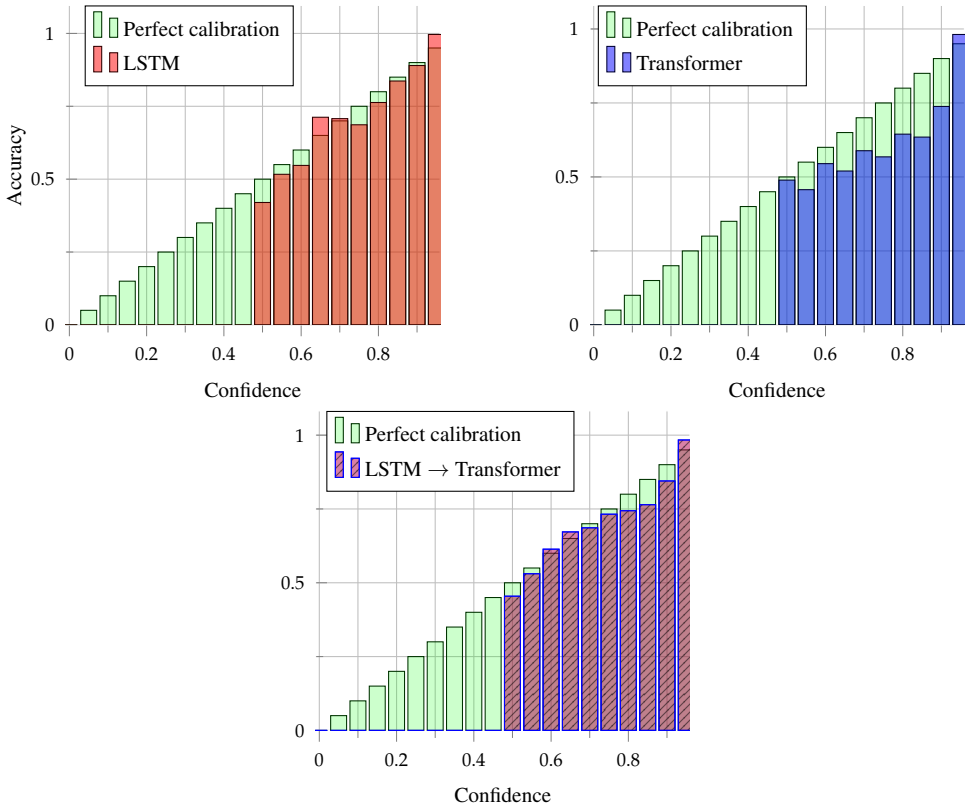
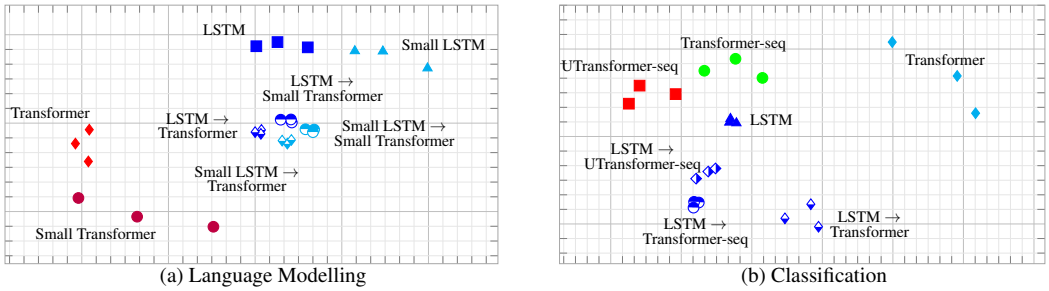


Figure 7.3: Calibration plots for independent and distilled Transformer for the classification setup. Note that since the task is binary classification, accuracy for confidences lower than 0.5 is not defined.

layer To compare and visualize the state of m different models to each other (at convergence or any stage of training), we propose using representational similarity [Abnar et al., 2019, Laakso and Cottrell, 2000] of the activations from their penultimate layer. Note that representational similarity measures how similar two models learn to represent the data in terms of the global "relations" between all the data points, not local example-by-example similarity. In fact, the "direct" similarity between the activations of the penultimate layers of two models can be quite low, while having high representational similarity. This is because models can keep the relations between data points similar while embedding data into completely different representational spaces. This is particularly useful when these models do not have the same architecture and their parameter space is not directly comparable. To do so, given a sample set of size n from the validation/test set (e.g. 1000 examples), we feed them to the forward pass of each model to obtain the representation from the penultimate layer of the models. Then, for each model, we calculate the similarity of the representations of all pairs from the sample set using dot product which leads to a matrix of size $n \times n$. We use the samples similarity matrix associated with each model to compute the similarity between all pairs of models. To do this, we compute the dot product of the corresponding rows of these two matrices after normalization and average all the similarities of all rows, which leads to a single scalar. Given all possible pairs of models, we then have a model similarity



(a) Language Modelling (b) Classification

Figure 7.4: 2D projection of representational similarity of the activations from the penultimate layers for 1000 examples from the validation set (check Appendix 7.3.2 for more details). We use the notation of $a \rightarrow b$ to refer to the student model b distilled from teacher model a.

matrix of size $m \times m$. We then apply a multidimensional scaling algorithm[2] to embed all the models in a 2D space based on their similarities.

Figure 7.4 shows the 2D projection of the relational similarity of representations[3] [Laakso and Cottrell, 2000] from the penultimate layer of the models. We see that, in the LM setup, the internal representations of student Transformers that are distilled from LSTMs are structured differently compared to independent Transformers and are more similar to the LSTM models. For the classification objective, we also see that the distilled models are further away from their independent versions. This supports the idea that the effect of distillation goes beyond the output of the models and their final performances.

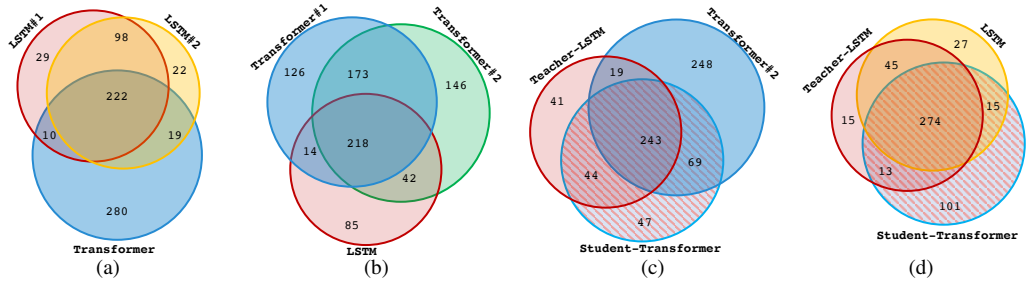
7.3.3 Per-sample Behaviour

To compare the models with each other and better understand how distillation affects the student models, we take a closer look at their per sample behavior and investigate if the errors a student model makes are more similar to its teacher's errors. Here, we look into the error overlap of the students and teachers, which reflects their similarity in terms of their behavior per data example. This similarity can be another proxy to measure the similarity of the solutions learned by the models, with and without distillation. Figures 7.5, 7.6, and 7.7 illustrates the error overlap between different models as Venn diagrams when they are trained independently and when we use distillation.

In Figure 7.5, we observe that when the Transformer and LSTM models are trained independently, two independent LSTMs behave more similarly compared to two Transformers (Figures 7.5b and 7.5a). Given a similar number of trainable parameters, i.e., similar capacity for LSTMs and Transformers, this again supports the claim that models with stronger inductive biases converge to more similar solutions (Also shown in Figure 6.4a).

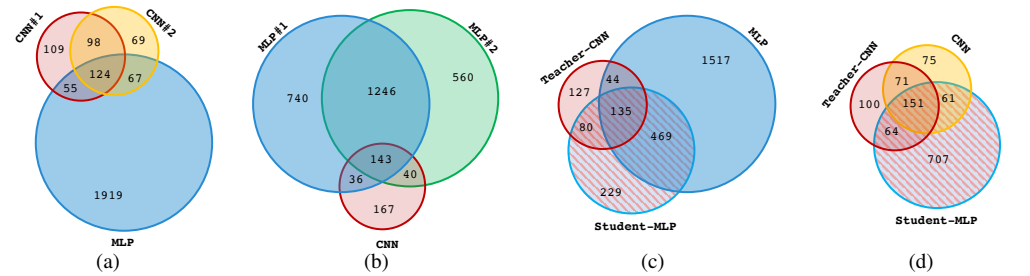
[2] *https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html*

[3] Note that the relational similarity captures the similarity of the structures, not the absolute values.



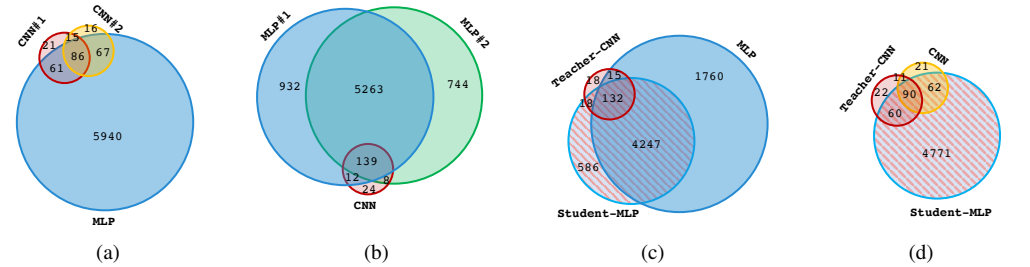
(a) (b) (c) (d)

Figure 7.5: Error overlap for LSTM and Transformer models trained with the classification objective on SVA task. These Venn diagrams show the intersections of the sets of examples miss-classified by the models. In (a) we compare two independent LSTMs (LSTM#1 and LSTM#2) and an independent Transformer; in (b) we compare two independent Transformers (Transformer#1 and Transformer#2) and an independent LSTM; in (c) we compare a student Transformer and a teacher LSTM with an independent Transformer; and in (d) we compare a student Transformer and a teacher LSTM with an independent LSTM.



(a) (b) (c) (d)

Figure 7.6: Error overlap for CNN and MLP models trained on MNIST and tested on Scaled-MNIST set from MNIST-C dataset. These Venn diagrams show the intersections of the sets of examples miss-classified by the models. In (a) we compare two independent CNN (CNN#1 and CNN#2) and an independent MLP; in (b) we compare two independent MLP (MLP#1 and MLP#2) and an independent CNN; in (c) we compare a student MLP and a teacher CNN with an independent MLP; and in (d) we compare a student MLP and a teacher CNN with an independent CNN.



(a) (b) (c) (d)

Figure 7.7: Error overlap for CNN and MLP models trained on MNIST and tested on Translated-MNIST set from MNIST-C dataset. These Venn diagrams show the intersections of the sets of examples miss-classified by the models. In (a) we compare two independent CNN (CNN#1 and CNN#2) and an independent MLP; in (b) we compare two independent MLP (MLP#1 and MLP#2) and an independent CNN; in (c) we compare a student MLP and a teacher CNN with an independent MLP; and in (d) we compare a student MLP and a teacher CNN with an independent CNN.

When we apply KD in a cross-architecture setting, with an LSTM teacher and a student Transformer, Figures 7.5d and Figure 7.5c, the student Transformer behaves more

Table 7.4: Performance (mean±std over 4 trials) of different LSTM and Transformer models trained independently with the LM objective on the training set.

Model	Perplexity ↓	$\mathcal{D}-$Accuracy ↑	$\mathcal{A}-$Accuracy ↑
Transformer	29.62 ± 0.10	0.956 ± 0.001	0.936 ± 0.004
Small Transformer	33.02 ± 0.05	0.959 ± 0.001	0.948 ± 0.005
LSTM	28.92 ± 0.08	0.964 ± 0.003	0.955 ± 0.003
Small LSTM	31.03 ± 0.11	0.964 ± 0.001	0.952 ± 0.006

similarly to the LSTM teacher and an independent LSTM, compared to the independent version of itself. This confirms that through distillation the way the student model solves the task becomes more similar to the way the teacher model solves the task.

We have similar observations in Figures 7.6, and 7.7; where errors of a student MLP are less and more similar to the errors the teacher CNN compared to an independently trained MLP.

7.3.4 Performance Scores on the Training Data

In the paper, for our first test case, we report the performance of LSTM and different Transformer models on the test set, when trained independently and with knowledge distillation. We observe that LSTMs achieve better accuracy on the test set compared to Transformers due to their inductive biases. Here, we also report the performance of all the models, for both classification and LM setup, on the training set, which confirms that Transformer models have enough capacity to achieve good scores on the training data.

This solidifies the narrative that the inductive bias of LSTMs is helping with generalization and rules out, for example, the possibility that LSTMs have a higher capacity or are trained better.

Table 7.5: Performance (mean±std over 4 trials) of different LSTM and Transformer models trained independently with the classification objective on the training set.

Model	Train $\mu-$Accuracy ↑
Transformer	99.57
Transformer-seq	99.57
UniversalTransformer-seq	99.66
LSTM	98.62

7.4 Distilling CNNs into MLPs

To evaluate the robustness of our findings on the transfer of inductive biases through KD, we performed a second case study, using different neural architectures and a different

task. We use convolutional neural networks (CNN) vs. multilayer perceptrons (MLP) as two families of models with different inductive biases. CNNs are the de facto choice for processing data with grid-like topology. Sparse connectivity and parameter sharing in CNNs make them an effective and statistically efficient architecture. The particular form of parameter sharing in the convolution operation makes CNNs equivariant to translation [Goodfellow et al., 2016]. Note that, we can view CNNs as MLPs with an infinitely strong prior over their weights, which says that first of all the weights for each hidden unit are identical to the weights of its neighbor with a shift in space, second, the weights out of the spatially continues receptive field assigned to each hidden unit are zero.

7.4.1 Models Architectures and Training Setup

We study CNNs and MLPs in the context of the Corrupted-MNIST dataset (MNIST-C) [Mu and Gilmer, 2019], which aims at benchmarking out-of-distribution robustness. We train the models on the original MNIST training set and evaluate them on the Translated and Scaled MNIST test sets from MNIST-C. In this scenario, the inductive biases of CNNs help them generalize better than MLPs.

Our CNN architecture is a stack of convolutions and pooling layers. Combining convolution and pooling over spatial regions results in invariance to translation. To have CNNs that can learn to be invariant to other transformations like changes in the scale, we can use cross-channel pooling [Goodfellow et al., 2013], where we pool over separately parametrized convolutions that have learned to detect different transformed versions of the same underlying features. Our MLP is simply a stack of fully-connected layers.

For training the independent models we use the Adam optimizer [Kingma and Ba, 2014] with exponential decay learning rate scheduler and for the student models in the distillation process, we use Adam optimizer with cosine decay restart [Loshchilov and Hutter, 2017] learning rate scheduler. The hyperparameters related to the regularization and learning rate schedulers are tuned separately for each model/experiment. For each model, we report the set of hyper-parameters that gives the best average performance across multiple trials with different random seeds for initialization.

7.4.2 On the Importance of Translation Equivariance.

Table 7.6 presents the accuracy and ECE of CNNs and MLPs when trained independently. All models are trained on the original MNIST training set and tested on the *Scaled* and *Translated* sets from MNIST-C. Even though CNNs' accuracy and ECE on the original MNIST test set are only slightly better than MLPs (.992 vs .985), there is a rather large gap between their performances on the Scaled (.962 vs. .794) and Translated (.981 vs. .373) test sets. This is expected since the inductive biases of CNNs

Table 7.6: Accuracy and Expected Calibration Error (mean±std over 4 trials) of CNN and MLP trained independently on MNIST and evaluated on MNIST, MNIST-Scaled and MNIST-Translated.

(a) Accuracy

Model	MNIST	Scaled	Translated
CNN	**0.992** ± 0.0009	**0.962** ± 0.0021	**0.981** ± 0.0003
MLP	0.985 ± 0.0011	0.794 ± 0.0154	0.373 ± 0.0151

(b) Expected Calibration Error

Model	MNIST	Scaled	Translated
CNN	**0.011** ± 0.0006	**0.060** ± 0.0044	**0.028** ± 0.0016
MLP	0.015 ± 0.0006	0.175 ± 0.0081	0.564 ± 0.0091

Table 7.7: Accuracy and Expected Calibration Error (mean±std over 4 trials) of CNN and MLP trained with pure distillation with $\tau = 5$, on MNIST and evaluated on MNIST, MNIST-Scaled and MNIST-Translated.

(a) Accuracy

Student Model	MNIST		Scaled		Translated	
	CNN	**MLP**	**CNN**	**MLP**	**CNN**	**MLP**
CNN	**0.991** ± 0.0004	**0.990** ± 0.0007	**0.951** ± 0.0046	**0.955** ± 0.0065	**0.978** ± 0.0003	**0.976** ± 0.0012
MLP	0.988 ± 0.0005	0.985 ± 0.0015	0.904 ± 0.0073	0.839 ± 0.0096	0.510 ± 0.0148	0.395 ± 0.0069

(b) Expected Calibration Error

Student Model	MNIST		Scaled		Translated	
	CNN	**MLP**	**CNN**	**MLP**	**CNN**	**MLP**
CNN	0.014 ± 0.0004	**0.013** ± 0.0005	**0.068** ± 0.0043	**0.054** ± 0.0063	**0.033** ± 0.0006	**0.030** ± 0.0016
MLP	**0.013** ± 0.0004	0.015 ± 0.0012	0.109 ± 0.0053	0.155 ± 0.0079	0.432 ± 0.0136	0.555 ± 0.0038

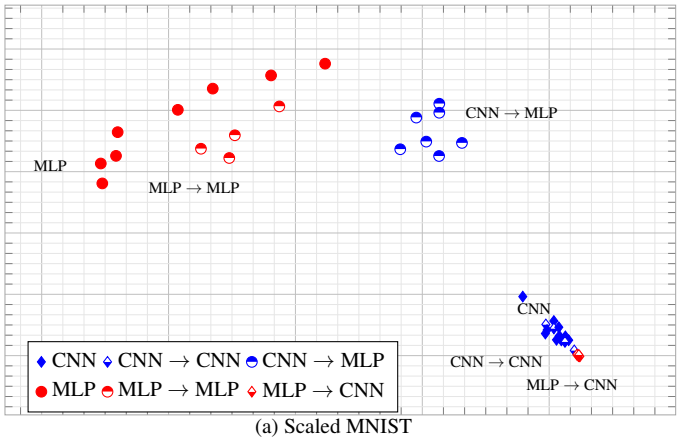
make them suitable for these types of generalizations. Moreover, the variance of the results from the CNNs is much less compared to MLPs. This is due to the fact that different instances of a model with stronger inductive biases are more likely to converge to solutions that belong to the same basin in the loss landscape [Neyshabur et al., 2020] (See §7.6 for more analysis on the relation of solutions different models converge to in the loss landscape).

7.4.3 Better Out of Distribution Generalization with KD.

Table 7.7 shows that distillation from a CNN into an MLP improves both accuracy and ECE for all three test sets, decreasing the gap for the Scaled test set (.904 vs. .794 without KD), and much more improvement on the performance on the Translated test set (.510 vs. .373 without KD). We also see a lower variance in the performance of MLP models that are trained through KD with CNN teachers.

We further compare the results of all possible pairs of models as teachers and students, to take into account different effects of KD that can potentially improve the performance of the student model. Although self-distillation results in a slightly better performance in MLPs, perhaps due to the regularization effect of distillation [Mobahi et al., 2020, Tang et al., 2020], the improvement in the performance of MLPs with an MLP teacher is much less compared to when the teacher is a CNN. Regardless of the teacher (MLP or CNN), KD results in slightly lower performances in student CNNs compared to CNNs trained independently (similar to results of an LSTM student in test case 1).

Furthermore, in Figure 7.8, we compare the relational similarity of the representations from penultimate layers of independently trained CNNs and MLPs as well as their



(a) Scaled MNIST

(b) Translated MNIST

Figure 7.8: 2D projection of representational similarity of the activations from the penultimate layers for all examples from the test set (check Appendix 7.3.2 for more details). We use the notation of $a \rightarrow b$ to refer to the student model b distilled from teacher model a.

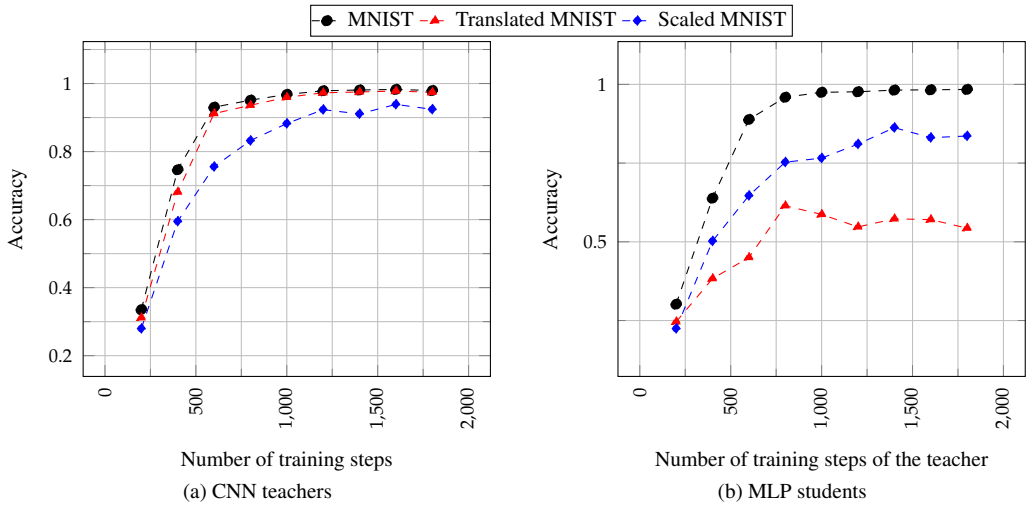
distilled ones. First of all, as expected based on our assumptions about the inductive biases of these models, MLPs have more variance than CNNs. Second, distilling from a CNN to an MLP results in representations that are more similar to the representations learned by CNNs, while this is not the case with MLPs as teachers and CNNs as students. Moreover, for both CNNs and MLPs, self-distillation does not significantly change the representations they learn.

Finally, we compare the paths the models follow during training until they converge to a solution. To plot the training path of a model, we compute the pairwise representational similarity between different stages of training of the model. Figure 7.1, illustrates the training path for an independent MLP, an independent CNN, and an MLP that is distilled from a CNN. While MLP and CNN seem to have very different behavior during training, the student MLP with a CNN as its teacher behaves differently than an independent MLP and more similar to its teacher CNN. This is interesting, in particular, since the student model is only exposed to the final solution the teacher has converged to and no information about the intermediate stages of training is provided in the offline KD.

7.4.4 Impact of the Quality of the Teacher

Here, in an ablation experiment for our second case study, we investigate the impact of the quality of the teacher in the in-distribution set on the generalization of the student in the out-of-distribution set. To do so, given a CNN as the teacher and an MLP as the student, we take snapshots of a CNN model during different stages of training as teachers with different qualities (we use 9 different teachers). Using each teacher, we train an MLP student.

Figure 7.9a presents the quality of the different teachers based on different test sets: Vanilla MNIST (in-distribution), Translated MNIST (out-of-distribution), and Scaled MNISt (out-of-distribution). For the CNN models that are trained with ground truth labels on vanilla MNIST, as expected, as the number of training iterations grows, the performance of the model on all three test sets increases. In Figure 7.9b, we see that in general, the accuracy of the MLP students follows the same trend, i.e., a better CNN teacher results in a better MLP student. Given the results of an independently trained MLP from Table 7.6a, the benefit of training an MLP via distillation for better generalization on in and out of distribution sets only kicks in when we have a CNN teacher with a quality more than a certain threshold.



(a) CNN teachers (b) MLP students

Figure 7.9: Effect of the quality of the teacher CNNs on the accuracy of the student MLPs. In the left plot, points that share the value on the x-axis represent the quality of a CNN, with respect to different test sets: Vanilla MNIST (in-distribution), Translated MNIST (out-of-distribution), and Scaled MNISt (out-of-distribution). In the right plot, similar to the left plot, points with the same x-value represent the quality of a same MLP model, trained via KD using the teacher on the corresponding place in the left plot, evaluated on the Vanilla, Translated, and Scaled MNIST test sets.

7.4.5 Impact of the Dataset Used in the Distillation Step

In our experiments in this paper, our focus is on the setups where we use the same dataset that was used to train the teacher model, to transfer its knowledge to the student

model.

We use this setup mainly because we want to see how effective is the distillation process to transfer the generalization behavior of the teacher in isolation, as using a different dataset in the distillation step would add another factor. In other words, during the training of the teachers and as well as the students (i.e., distillation step), we only use samples from the in-distribution set to make sure the desired generalization behavior is not apparent from the dataset used for training neither the teacher nor the student models.

In this section, we extend the CNN-MLP experiments on Corrupted-MNIST and look into the performance of the student model, when we use samples from the out-of-distribution set in the distillation step for training the student.

Table 7.8: Accuracy of MLPs trained through KD with CNN teachers, where the CNN teachers are trained on in-distribution (vanilla MNIST) training set, while the training set in the distillation step is either in-distortion (first row) or out-of-distribution (second and third rows). Note that during the distillation step, the student do not have access to the ground truth labels from the training set.).

Distillation Dataset	Test Dataset		
	MNIST	Translated MNIST	Scaled MNIST
MNIST	0.99 ± 0.001	0.51 ± 0.015	0.90 ± 0.007
Translated MNIST	0.79 ± 0.015	0.98 ± 0.001	0.53 ± 0.015
Scaled MNIST	0.79 ± 0.016	0.30 ± 0.009	0.98 ± 0.001

Table 7.9: Accuracy of MLPs trained with ground truth labels on different splits of the Corrupted-MNIST dataset.

Training Dataset	Test Dataset		
	MNIST	Translated MNIST	Scaled MNIST
MNIST	0.99 ± 0.001	0.37 ± 0.015	0.79 ± 0.015
Translated MNIST	0.62 ± 0.011	0.98 ± 0.001	0.42 ± 0.039
Scaled MNIST	0.76 ± 0.014	0.26 ± 0.006	0.99 ± 0.001

Table 7.10: Accuracy of CNNs trained with ground truth labels on different splits of the Corrupted-MNIST dataset.

Training Dataset	Test Dataset		
	MNIST	Translated MNIST	Scaled MNIST
MNIST	0.99 ± 0.001	0.98 ± 0.000	0.96 ± 0.002
Translated MNIST	0.99 ± 0.001	0.99 ± 0.001	0.97 ± 0.002
Scaled MNIST	0.88 ± 0.010	0.88 ± 0.014	0.99 ± 0.001

Table 7.8 presents the result of an MLP student when we use different training sets in the distillation step. We can see that when distilling knowledge from a CNN teacher that is trained on vanilla MNIST, if we use translated or scaled MNIST in the distillation step, the student MLPs achieve relatively high performance on the corresponding test sets, while the performance on the other out-of-distribution set drops compared to when

we use vanilla MNIST in the distillation step. To have complementary information for better comparisons, Table 7.9 shows the accuracies of MLPs when they are directly trained on each of these datasets. Interestingly, we observe that when trained through KD the performances of the student MLPs are higher or match the performance of MLPs when trained with ground truth labels, and additionally, they achieve better performance on the other datasets. For example, in the case when we use translated MNIST to train the MLPs, the accuracies of the student MLPs match the accuracy of MLPs trained with the ground labels, while the accuracies on the other two datasets (vanilla and scaled) are higher for the student MLPs trained with CNN teachers.

7.5 Persistency of the Transferred Effects of Inductive Biases

We run an ablation study to show whether the qualitative aspects of the solutions the student models converge to that are rooted in the inductive biases of their teacher, persist even if we stop distilling from the teacher model and expose the student to the ground-truth labels. If the student model has converged to a solution, which is optimum with respect to the ground-truth loss as well as the distillation loss, we would expect the solution to be more or less stable. On the other hand, the loss landscape could be inherently different for the distillation loss compared to the ground-truth loss.

To understand to what extent the effects of KD persist after the student model is exposed to ground-truth labels, we compare the performance of student models trained with two strategies: pure distillation and scheduled distillation. In pure distillation, the student is only trained with the distillation loss, whereas in scheduled distillation the student model is trained with a mixture of distillation loss and ground-truth loss, where the ratios of these two terms changes during training with respect to some scheduling. Here we employ the simple zero-one schedule. We train the student model with pure distillation for a specific number of steps, and then switch the optimization objective to pure ground-truth loss. Additionally, we investigate whether the persistency of the effects is different for offline distillation and online distillation. In offline distillation, which is the standard setup for KD, the teacher model is fully trained beforehand and its parameters are frizzed during the distillation process, whereas in online distillation the student and the teacher are trained simultaneously. Our hypothesis is that exposing the student model to the training path of the teacher model in online distillation could lead to more stable effects.

We look into the persistence of the effects for the two settings of a Transformer model trained on language modelling with an LSTM teacher, Figure 7.10 and 7.11, MLP model trained on MNIST digit classification with a CNN teacher, Figures 7.12 and 7.13.

In the language modelling setup, we obtain lower perplexity, which is the ground

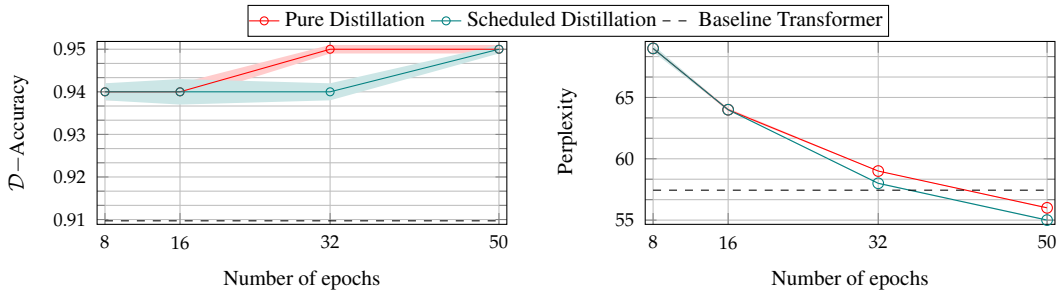


Figure 7.10: Persistence of the effects of offline knowledge distillation from LSTM to Transformer in the language modelling task. We get slightly better perplexity with scheduled distillation, while the accuracy of the model on subject-verb agreement drops compared to pure distillation. With respect to both metrics, perplexity and accuracy on subjecy-verb agreement task, the performance of the model trained through scheduled distillation is better than its stand alone performance.

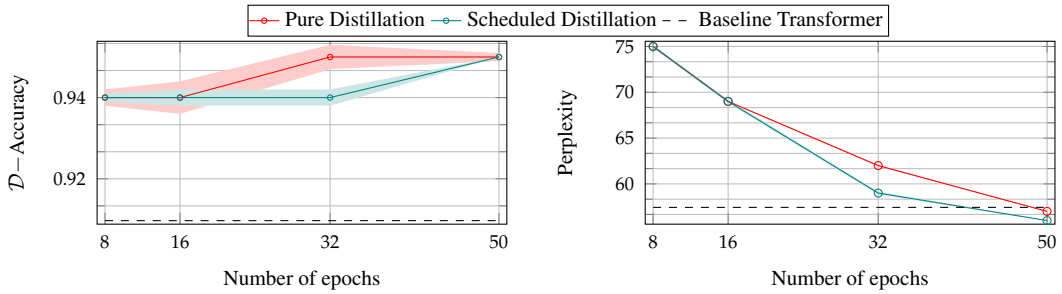
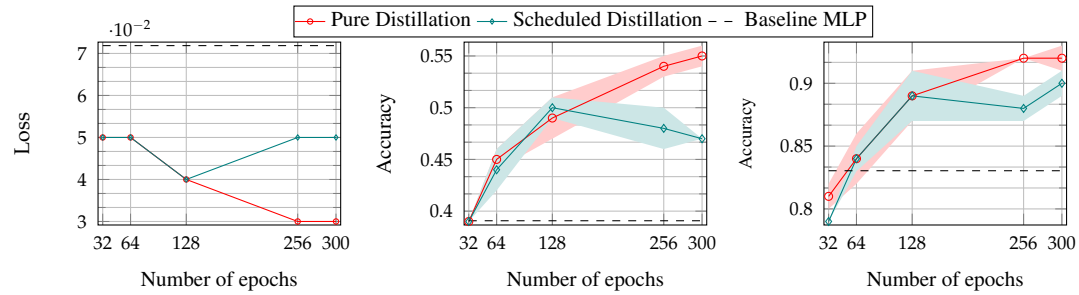


Figure 7.11: Persistence of the effects of online knowledge distillation from LSTM to Transformer in the language modelling task. We get slightly better perplexity with scheduled distillation, while the accuracy of the model on subject-verb agreement drops compared to pure distillation. With respect to both metrics, perplexity and accuracy on subjecy-verb agreement task, the performance of the model trained through scheduled distillation is better than its stand alone performance.

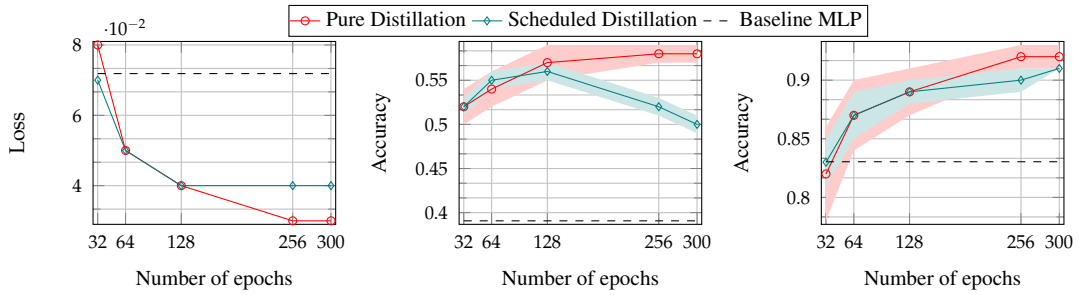


(a) MNIST (b) Translated MNIST (c) Scaled MNIST

Figure 7.12: Persistence of the effects of offline knowledge distillation from CNN to MLP in MNIST classification task.

truth training objective, with scheduled distillation compared to both pure distillation and independently trained Transformer, while the accuracy on the SV-agreement task remains the same. This could mean that even though the student model converges to a different solution upon training with ground-truth labels, the benefits of initial KD do not completely fade away in this case.

In the MNIST experiments, we observe an overfitting issue as the loss of the model on



(a) MNIST (b) Translated MNIST (c) Scaled MNIST

Figure 7.13: Persistence of the effects of online knowledge distillation from CNN to MLP in MNIST classification task.

the in-distribution test set increases slightly, however, the performance of the model on the out-of-distribution test set remains higher than an MLP model trained independently. Interestingly, in this case, we observe that for online KD the divergence between the performance of the models trained with pure distillation and scheduled distillation is less compared to offline KD.

Initializing a model through KD can be beneficial, but not necessarily. While the persistence of the effects of inductive biases of the teacher model on the solution that the student model converges to can not be guaranteed once we start to train the student model with other supervision signals, e.g., ground-truth labels, in some cases, the benefits could be preserved to some extent. Additionally, it seems the distillation process itself determines to what extent the effects are persistent. In our experiments, we find the gains in performance through online distillation to be more stable than the gains in performance through offline distillation.

7.6 Do the distilled models converge to the same basin in the loss landscape?

To gain a better understanding of the effect of KD and inductive biases of the models from an optimization point of view, we looked into how different models relate in terms of the solutions they converged to in the loss landscape.

To do so, inspired by the discussion in [Neyshabur et al., 2020], we look into different pairs of models and check if their final solution belongs to the same flat basin[4] of the loss landscape or they converged to completely different optima. To do so, given two models,m_1 and m_2, we take their parameters, θ_1 and θ_2, and evaluate a series of models obtained by linearly interpolating θ_1 and θ_2, with different coefficient, i.e., the parameters of model m_i is computed as $\theta_i = \lambda_i\theta_1 + (1-\lambda_i)\theta_2$.

It has been shown [Neyshabur et al., 2020] that if the converged solutions of m_1 and

[4] Basin refers to areas in the parameter space where the loss function has relatively low values.

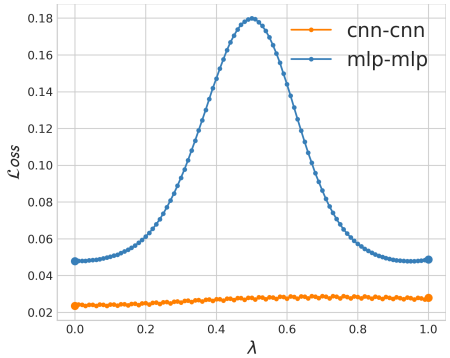


Figure 7.14: Performance barriers between different instances of MLPs and CNNs (with the same initialization), in terms of loss on the test.

m_2 belong to the same flat basin of the loss landscape, the models obtained by linearly interpolating their parameters are well-behaved because they also remain in that basin. However, for two models that converge to different optima and don't share the flat basin of the loss landscape, the liner interpolations do not lead to well behave models. Here well behaved means performing well on the training task.

Here, we first, compare different instances of MLPs and CNNs. We train two instances of the same architecture with the same initial state but different random seeds (which would lead to different ordering of training examples, and different dropouts). Figure 7.14 shows the loss on the test set (y axis) for the two trained instances, as well as models obtained by linear interpolation of the two models with different λs (x axis). In the case of MLPs, there is a large barrier between the two instances, showing that these models, even with the same initialization, will converge to solutions in different basins of the loss landscape. In contrast, for CNNs, their strong inductive biases drive them to converge to the solutions in the same basin, regardless of the stochasticity of the training process. This also supports the higher variance in the results we report for models with weaker inductive biases in §7.3.2 and §7.4.3.

Next, we look into the effect of distillation on the diversity of the basins different instances of models converge to. Figure 7.15 shows the performance barriers of different pairs of MLPs (MLP#1 and MLP#2), when they are trained independently (i.e. when the teacher is data), as well as trained through KD, with an MLP and a CNN model as teachers.

First of all, we observe that two models, initialized similarly but with different random seeds, trained through distillation with the same teacher are likely to converge to the same area in the loss surface (plots (c) and (f)). This happens regardless of the inductive bias of the teacher and student models. Comparing the plots in the diagonal of Figure 7.15, we can see that for both $CNN \rightarrow MLP$ (plot f) and $MLP \rightarrow MLP$ (plot c) the performance barrier is rather small in contrast to the large barrier between two independently trained MLPs (plot a). This indicates the power of KD to narrow down the search space of the student model and drive it to a particular set of solutions.

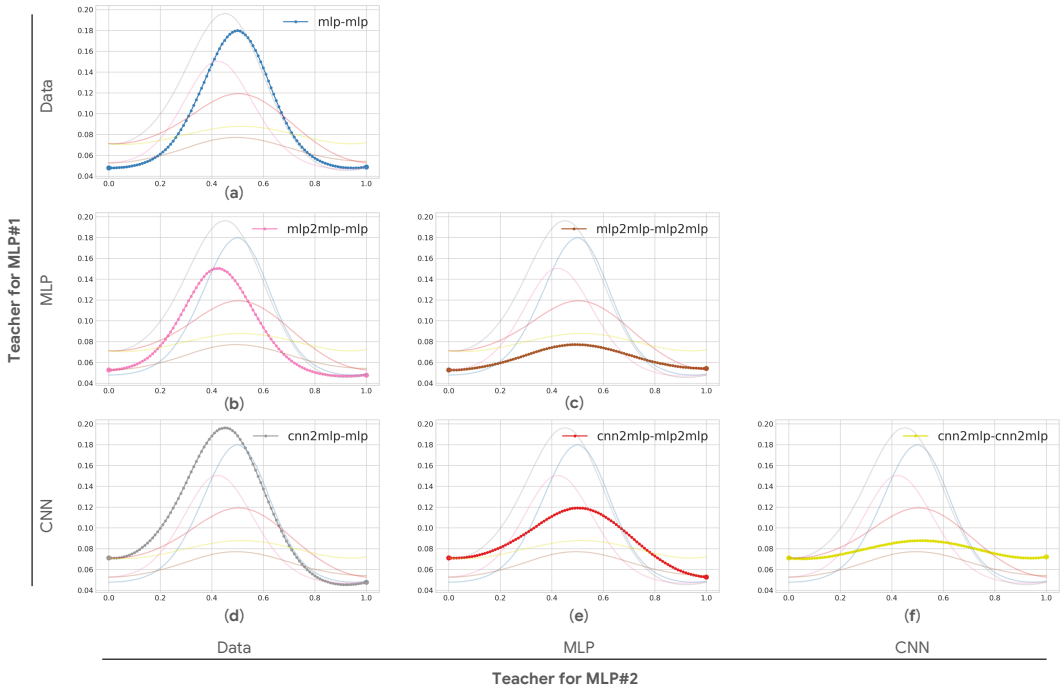


Figure 7.15: Performance barriers between different instances of MLPs with the same initialization trained independently or through knowledge distillation. Here y-axis on each subplot is the value of the loss on the test set and the x-axis is the value of the interpolation coefficient, λ. The rows in the figure correspond to the teacher of the instance on the left side (MLP#1) and the columns correspond to the teacher of the instance on the right side of the plots (MLP#2).

Moreover, comparing the distilled instance of a model with an independently trained instance with the same initialization and different random seeds, the first column of Figure 7.15 (plots (a), (b), and (d)), we see that the distilled instances and independent instances are not in the same basin, regardless of the teacher but the barrier is larger (larger bump in the plots) when the teacher has a stronger inductive bias ($CNN \rightarrow MLP$). Similarly, as depicted in the second and third columns of Figure 7.15, while models distilled from the same teacher seem to be close in the loss surface (plots (c) and (f)), models distilled from different teachers (plot (e)) seem to be further away (have a larger barrier in between).

7.7 Conclusion

In this chapter, we investigate the inductive biases and expressive power of neural networks through the lens of knowledge distillation. In addition to illustrating how KD can be used to shed light on differences among different models, we investigate to what extent the effects of inductive biases of the teacher models on its solution, during training and at the converged point, transfers to the student model through

knowledge distillation. Findings from our experiments provide insights about some aspects of why and when knowledge distillation can be beneficial as a way to increase the generalization performance of models, or more generally to take advantage of inductive biases of different models at the same time.

First, we demonstrate how inductive biases arising from different architectural choices affect the generalization behavior of the models we study. We further show that when a model has the right inductive bias to learn a task in a generalizable manner, we can transfer its knowledge to a model that lacks the needed inductive bias. We show that solutions that the student model learns are not only quantitatively but also qualitatively reflecting the effects of the inductive biases of the teacher model.

In chapter 6, we demonstrated how different sources of the recurrent inductive bias impact the performance of the models by progressively incorporating them into Transformer architectures. Following this, in this chapter using the same task and similar experimental setup, we show the effects of the recurrent inductive bias of LSTM on its solution transfer to a standard transformer when we train the Transformer through knowledge distillation, shrinking the gap between the performance of these models.

Additionally, we show that the persistence of these effects upon exposing the student model to the ground-truth labels in the training data is not guaranteed and depends on multiple factors, such as the task and the distillation process. While in some cases initializing a model through KD from a model with different inductive biases and continuing the training with ground-truth labels can be beneficial, in another case the student model could potentially diverge to a different solution and might eventually forget the effects of the inductive biases of the teacher.

In most of our experiments, our focus is on offline distillation using the commonly used cross-entropy loss. We also consider a case of inline distillation when studying the persistency of the effects of the distillation process. Generally, we recognize that the details of the distillation process itself might have a major impact on its effects. Hence, a next step is to look into different distillation strategies, such as online distillation [Anil et al., 2018], relational KD [Park et al., 2019], or similarity preserving KD [Tung and Mori, 2019], to better understand their effectiveness for transferring the effect of inductive biases. Another aspect of the distillation process that can be further investigated is when we have multiple teachers, each with different inductive biases that are useful for different tasks. How would the effects of the inductive biases from teachers interact in the solution space of the student model? In these settings would we be able to guide the student toward a more generalized solution than the teachers which hopefully combines all their benefits in one model?

PART

IV

Conclusion

7.8 In Search of Inductive Biases

At the time we started to work on this thesis, in different domains, specifically NLP, we were observing a new line of general-purpose models succeeding in solving a variety of tasks. It all started with word embedding models based on shallow neural network architectures [Mikolov et al., 2013, Pennington et al., 2014]. These models were designed to compute representations of tokens/words, which were then plugged into other machine learning models to solve different NLP tasks. These efforts were then followed by using more sophisticated neural network architectures such as variants of RNNs that could process longer pieces of linguistic input and solve tasks in an end-to-end manner or learn contextual representations of words/tokens.

At the time, the main focus of research on neural networks was to build task-specific architectures with the right inductive biases that can learn to solve the given tasks with reasonable amounts of data. For example, neural networks with convolutional and pooling layers that make them translation and scale equivariant for processing images, recurrent and recursive neural networks for language processing, and various ways of combining different neural architectures to deal with more complicated tasks such as multi-modal question-answering systems.

Given the above-mentioned context, our main motivation in this thesis was to build techniques that would allow us to compare existing models beyond their final performance on the task they are trained on. We aimed to study the effect of different inductive biases and design choices beyond intuitions and speculations. We sought to understand if there is a real difference between different neural network language models and what explains their success and failures.

Most recently, a major factor in the success of deep learning models seems to be scale. Scaling up the dataset, model size and compute proportionally, we are now able to achieve much higher levels of performance and generalization on a large set of tasks than we ever could by hand-crafting models specifically for each task. While this progress might deem the research on inductive biases unnecessary at first glance, we argue otherwise.

In light of the increasing success of deep learning models in large-scale scenarios, general-purpose architectures have emerged and the main focus has shifted toward large-scale pretrained models that can transfer to any downstream tasks in few-shot or even zero-shot settings [Brown et al., 2020, Raffel et al., 2020, Ramesh et al., 2021, Saharia et al., 2022, Smith et al., 2022, Thoppilan et al., 2022]. Being able to learn from diverse and large amounts of data is the key to the impressive performance of these models. Hence, nowadays, rather than task-specific inductive biases, the main emphasis is on scalability [Rae et al., 2021]. I.e., finding neural network architectures that have enough capacity to learn from large amounts of data, while efficient and affordable in terms of memory and compute [Du et al., 2021, Shazeer et al., 2017].

The hope behind these efforts to scale is that by having sufficient data in quantity,

quality and diversity, these models can learn all the underlying rules that govern the data and generalize, without the need to have prior assumptions about the desired generalization behaviours. As appealing as this idea sounds, taking the big picture of recent works into account, even large-scale models would not automatically, consistently, generalize to out-of-distribution data [Abnar et al., 2022, D'Amour et al., 2020]. Here by automatically, we mean without any built-in assumptions or prior about how they should generalize. Intuitively, one of the roots of this challenge is the difficulty of encoding all the desired generalization behaviours in the data (the under-specification problem) [D'Amour et al., 2020].

There is no doubt that we need a minimal set of inductive biases to ensure the models learn in a generalizable manner that is consistent with our expectations, rather than memorizing all the data points independently or generalizing in arbitrary and non-meaning-full manners. Even in the infinite data regime, there will always be gaps in the data distribution where the model needs to interpolate or extrapolate, and these inductive biases determine the interpolation and extrapolation mechanisms. We need to identify what are these biases and how are/can they be incorporated into neural network models. One challenge that we need to be aware of is that it might not be possible to identify the crucial inductive biases independent of the scale of data available to the model.

Furthermore, to be able to predict the performance of the models on new tasks and to be able to rely on them in out-of-distribution settings, we need to understand their inductive biases. In other words, developing techniques to study inductive biases of the models helps us to predict different aspects of their performance in different settings and on different data distributions.

Ultimately, we hope to discover inductive biases that would make the models more efficient in terms of data and compute and at the same time more reliable in OOD settings. This would not only be beneficial for practical uses of ML models but also a step toward a better understanding of human intelligence, which is the most scalable and efficient learning algorithm that we know of.

7.9 Summary of Contributions

In this thesis, we investigated different techniques to shed light on how neural networks with presumably different inductive biases process language. We argued that it is crucial to evaluate these models from a variety of perspectives, rather than just their final performance on a given task.

Given two learning algorithms, comparing their performances on a given set of tasks is the most straightforward solution to study their differences. But often, when considering a single task, there is more than one solution that achieves a certain level of performance or even completely solves it. An example of this, which has been studied in the context

of neural networks, is short-cut learning [Geirhos et al., 2020]. For example, when a model learns to classify objects relying on spurious features that are consistently correlated with the causal features across the training data. In other words, if a learning algorithm achieves a human-level performance on a task we can not necessarily conclude that it is solving it in the same manner humans do.

Hence, to deeply understand how different are different models, we need to widen our comparison framework by: (1) including the wide range of tasks and domains in our evaluation set that can reveal different aspects of the learned solution. (2) Doing a thorough analysis to understand the characteristics of the learned solution.

In this thesis, we mainly sought evaluation and analysis techniques that can help us shed light on the underlying processes in various learning algorithms, and demonstrate the impact of different inductive biases on the solutions these models converged to. In chapter 2 and 3, we introduced and applied new techniques for understanding how neural network models, specifically trained to model language, work.

In chapter 2, we introduced ReStA (Representational Stability Analysis) to study the sensitivity of the representational spaces of the models to different factors (conditions) [Abnar et al., 2019]. We used ReStA to study the effect of context, context length, or depth on the representations obtained from neural language models.

Our analysis of the context sensitivity of different layers of different neural network-based language models sheds light on different strategies they have for integrating contextual information. In particular, we observed that in recurrent neural network language models, the top layer is more sensitive to context length compared to the first layer. This could be an indication that this model has formed a hierarchical underlying mechanism for processing the sentences in these models.

In our analysis of how the representations evolve across layers of different neural language models, we observed that corresponding layers of different models, i.e., the i^{th} layers of two different models, are more similar compared to random layer pairs. We showed that indeed, different architectural choices, e.g., recurrence, parameter sharing in time or over spatial dimensions of the inputs, self-attention, size of the modes, and training objective can lead to very different representational spaces. Surprisingly, we find that in Transformer based language models some hyper-parameters such as depth and width do not have a significant impact on the general trajectory of representations as they evolve across layers.

Furthermore, in chapter 3, we introduced Attention Flow and Attention Rollout, to study information flow in neural network models with attention mechanisms (e.g., Transformers) [Abnar and Zuidema, 2020]. In a model with a self-attention mechanism, an intuitive approach to studying how information propagates across the layers is to investigate and visualize the attention patterns.

In the original Transformer models, representations at every layer are tied to input tokens and computed by mixing up the representations in the previous layer based

on the attention patterns. Prior to this work [Abnar and Zuidema, 2020], in many studies, raw attention patterns were used as indicators of the importance of different tokens of the input at each layer, ignoring the fact that the representations at higher layers do not necessarily represent their corresponding input token. Hence, there were a lot of criticisms about using attention patterns to explain the behaviour of these models [Grimsley et al., 2020, Jain and Wallace, 2019, Serrano and Smith, 2019].

By modelling/approximating the flow of signals across attention layers as a graph, we proposed simple techniques that translate attention to representations at each layer to attention to input tokens or attention to representation at any arbitrary layer in the network. The main idea behind these techniques is to take the mixing of information across layers into account. We showed, quantitatively, that interpreting attention weights in this manner has a stronger correlation with other input attribution methods than raw attention weights. Moreover, on a co-reference resolution task, we showed, qualitatively, that attention rollout and attention flow provide better explanations of the behaviour of the model.

Additionally, as a case study, we used attention rollout to understand the effective context length at different layers of different Transformer based language models. We showed that in all these models the effective context length grows as we move from the input layer towards the output layer. Based on our observations, among the models that we studied, the effective context length is impacted by the depth of the model, as well as other architectural details and the training objective.

Attention rollout and techniques built on top of that have been widely used as a post-hoc interpretation technique of attention-based models across vision and language domains. Furthermore, it has been shown that attention flow approximates Shapley values [Ethayarajh and Jurafsky, 2021, Metzger et al., 2022]. Shapely value is a concept in game theory to solve the credit assignment problem and in interpretable machine learning it is used as an input attribution method [Lundberg and Lee, 2017].

Besides our efforts to study and compare the effect of inductive biases of different neural language models, we tried to evaluate them based on their similarity to brain signals. In chapters 4 and 5, we built on top of prior work toward designing evaluation setups in which we can compare models with brain signals. The ultimate goal of such evaluation frameworks is to allow us to study the connection between the underlying mechanisms of language processing in humans and machines.

In our studies [Abnar et al., 2018, 2019], we found that, among existing neural network architectures, recurrence has a significant role in facilitating learning structures needed to solve language tasks more similar to the human brain. Our experiments indicated serious limitations in existing frameworks for using brain signals to evaluate computational learning algorithms which could question the scope and reliability of the outcomes of these experiments. One big challenge in this direction is the amount of available brain data in both controlled and uncontrolled settings, as well as ensuring that these signals contain information about the processes going on in the brain at different levels

of abstraction.

Considering our findings in part II and III, as well as the historical literature on neural network architectures, we decided that it would be fascinating to further investigate the inductive biases of recurrent neural networks. In a large-scale training regime, attention-based models achieve impressive performance. However, in small-scale training regimes, and on tasks that require learning hierarchical rules of language, the performance of RNNs is superior compared to Transformers. In addition, when comparing the internal states of these models to brain signals, LSTM(s) internal seemed to be more similar to human brain activations compared to Transformers.

To look further into the impact of recurrence, in chapter 6, we identified different sources of inductive biases in recurrent neural networks: (1) sequentiality, (2) memory bottleneck, and (3) parameter sharing in time. We injected each of these constraints gradually into a Transformer architecture to see how they each impact the final solution. Evaluating the resulting architectures on a task, designed to capture the ability of models to deal with long-range dependencies, we showed that indeed, each of these sources contributes to the success of recurrent neural networks in this task.

Finally, in the last chapter of this thesis, we touched upon the concepts of learnability and expressivity in neural networks. Consider a task and two models with different architectures, where one of them learns a more generalizable solution than the other. Is the reason behind the better performance, better inductive biases? Or does the lack of expressivity explain the poor performance? In other words, if we know a better solution exists but a model is not able to converge to that solution, how can we study if the limitation is in the expressivity of the model or the learnability of the solution for the model? Our initial motivation to work on this chapter was to empirically investigate this. We used knowledge distillation as a framework to investigate the benefits of inductive biases of different neural network architectures, and as a technique to empirically study their expressivity versus the learnability of different solutions for them.

More concretely, in chapter 7, we studied how the effect of inductive biases of different neural network architecture transfer through processes such as knowledge distillation [Abnar et al., 2020]. We observe that in certain settings, it is possible to transfer the effects of inductive biases of one model on different aspects of its performance, as well its representational space, to another through knowledge distillation. We demonstrate this in the case of Transformer and LSTM on a language modelling task, as well as for CNNs and simple MLPs on an image recognition task. Touvron et al. [2020] shows this for CNN-based models and Transformers on a larger scale. While vision Transformer models do not work well in small data regimes, Touvron et al. [2020] shows that we can achieve performance competitive to CNN-based models when we train them through distillation with CNN-based teachers. Additionally, we study the consistency of the transferred effects in models that are further trained directly on the data. We find that in some cases the effects of KD can persist even after continuing the training process with ground truth labels (instead of outputs of the teacher model).

These findings are important from different viewpoints: First of all, our experiments demonstrate how knowledge distillation can be viewed as a technique to test the strength of the inductive biases of the models and to study whether two models have similar or different inductive biases and generalize in the same manner.

Second, different models have different inductive biases and different advantages, while architecturally it might not be possible to have all the advantages in one model. Our findings show that it is possible to use KD to benefit from the inductive biases and advantages of different model architectures at the same time.

Finally, Knowledge distillation is a commonly used approach in practice to improve the performance of the models, and often as a compression technique. Given the fact that the teacher models can transfer their biases to the student models, we should be aware to avoid echoing unintended biases. For example, biases in the data that the teacher model is trained on.

7.10 Future Directions

Moving forward, it is crucial to extend the evaluation and interpretability toolkit in machine learning to allow us to compare the reasoning processes in neural networks and predict their generalization behaviour. While potentially the existing tools such as diagnostics classification and representational similarity analysis can reveal some aspects of the internal processes of these models, we are far away from having a systematic framework for comparing the decision processes of the models. Most of the recent work on trying to analyze and understand these models is focused on the final information captured in the representations learned by them.

Can we study neural networks with an algorithmic lens? Can we quantify the complexity of the algorithms a neural network model can execute? or the type of processes are executable by these models? How can we build upon the existing approaches to be able to answer questions like: “What are the sub-processes involved in processing the inputs in a given neural network?”, “What is the interaction between these processes?”, “How do they evolve during training?”, “How do the sub-processes that emerge during the training of a neural network and their interaction determine the generalization power of the neural network?”, “How do different inductive biases impact the emergence of different sub-processes?”.

As a concrete example of this, we can consider the mechanisms which allow models to deal with some forms of ambiguity. We can ask if existing neural network architectures can implement processes with feedback mechanisms. I.e., they follow a path toward a decision but can later adjust their representations and the path toward a different decision. Otherwise, an alternative mechanism would be to build up the graph for all possible decisions in a bottom-up manner and gradually prune this graph. Can we theoretically, or empirically show if a given model can implement either of these

mechanisms? Since we can not simply rely on the initial choices that we make in designing the neural networks and the training strategy to have a definite impact on their reasoning process or the final representations obtained from them, we need to further confirm these effects. How can we prob trained neural networks to investigate which type of process it relies on?

Another example of questions of this kind which is maybe more popular among NLP researchers is how neural network language models integrate different types of context. How can we identify the notions of long-term (what has been stored in the parameters of the model) and short-term (what is given to the model as part of the current input) memory as different types of context? And how can we prob the models to characterize and compare the context integration processes? Studies such as Chan et al. [2022] and Maheswaranathan and Sussillo [2020] investigate this to some extent but there are still many remaining open questions in this area.

Inspired by the studies on human cognition and human learning biases, we can form assumptions about the desired characteristics of the processes neural networks learn to execute. For example, modularity and how it related to the concept of cognitive flexibility in humans [Kim et al., 2012, Rikhye et al., 2018, Scott, 1962], or how different types of information are processed in a sequential or parallel manner in the human brain [Sigman and Dehaene, 2008]. Having these types of assumptions, we can directly probe neural networks searching for underlying processes with similar characteristics. Additionally, we can employ the frameworks for evaluating the connection between signals obtained from the human brain and representations obtained from neural networks to study alignments between the sub-processes or modules in the human brain and neural networks.

Having frameworks for systematic evaluation and comparison of neural networks to be able to answer questions such as the ones we raised above is just one of the primary steps. In light of the recent progress of large-scale pre-training approaches, we need these frameworks to be scalable. More precisely, we need evaluation frameworks and interpretation techniques that can deal with such large-scale models and datasets.

In the end, understanding how existing neural network models work and being able to predict their ability to generalize is not the ultimate goal. On the one hand, we want to be able to use these models to be able to indirectly probe the human brain and study language processing in humans. On the other hand, we seek to improve the performance and efficiency of these models in terms of both compute and data, perhaps guided by what we learn about language processing in the human brain.

Bibliography

Abnar, S., Ahmed, R., Mijnheer, M., and Zuidema, W. (2018). Experiential, distributional and dependency-based word embeddings have complementary roles in decoding brain activity. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 57–66. Association for Computational Linguistics.

Abnar, S., Beinborn, L., Choenni, R., and Zuidema, W. (2019). Blackbox meets blackbox: Representational similarity & stability analysis of neural language models and brains. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 191–203, Florence, Italy. Association for Computational Linguistics.

Abnar, S., Dehghani, M., Neyshabur, B., and Sedghi, H. (2022). Exploring the limits of large scale pre-training. In *International Conference on Learning Representations*.

Abnar, S., Dehghani, M., and Zuidema, W. (2020). Transferring inductive biases through knowledge distillation. *arXiv preprint arXiv:2006.00555*.

Abnar, S. and Zuidema, W. (2020). Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Ahmed, R. (2017). How the brain gives meaning to words. unpublished Bachelor thesis, Artificial Intelligence, University of Amsterdam.

Ahn, S., Hu, S. X., Damianou, A., Lawrence, N. D., and Dai, Z. (2019). Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'19.

Alhama, R. G. and Zuidema, W. (2019). A review of computational models of basic rule learning: The neural-symbolic debate and beyond. *Psychonomic bulletin & review*, 26(4):1174–1194.

Alishahi, A., Belinkov, Y., Chrupała, G., Hupkes, D., Pinter, Y., and Sajjad, H., editors (2020). *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, Online. Association for Computational Linguistics.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. (2019). *Gradient-Based Attribution Methods*, pages 169–191. Springer International Publishing.

Anderson, A. J., Zinszer, B. D., and Raizada, R. D. (2016). Representational similarity encoding for fMRI: Pattern-based synthesis to predict brain activity using stimulus-model-similarities. *NeuroImage*, 128:44–53.

Anil, R., Pereyra, G., Passos, A. T., Ormandi, R., Dahl, G., and Hinton, G. (2018). Large scale distributed neural network training through online distillation. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR'18.

Baddeley, A. (1992). Working memory: The interface between memory and cognition. *Journal of cognitive neuroscience*, 4(3):281–288.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *proceedings of the 2015 International Conference on Learning Representations*.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

Beinborn, L., Abnar, S., and Choenni, R. (2019). Robust evaluation of language-brain encoding experiments. *International Journal of Computational Linguistics and Applications*, to appear.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *arXiv:2004.05150*.

Bemis, D. K. and Pylkkänen, L. (2011). Simple composition: A magnetoencephalography investigation into the comprehension of minimal linguistic phrases. *Journal of Neuroscience*, 31(8):2801–2814.

Binder, J. R., Conant, L. L., Humphries, C. J., Fernandino, L., Simons, S. B., Aguilar, M., and Desai, R. H. (2016). Toward a brain-based componential semantic representation. *Cognitive neuropsychology*, 33(3-4):130–174.

Bingel, J., Barrett, M., and Søgaard, A. (2016). Extracting token-level signals of syntactic processing from fMRI - with an application to PoS induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 747–755. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

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

C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *arXiv*.

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

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06.

Buckner, R. L. (1998). Event-related fMRI and the hemodynamic response. *Human brain mapping*, 6(5-6):373–377.

Bulat, L., Clark, S., and Shutova, E. (2017). Speaking, seeing, understanding: Correlating semantic models with conceptual representation in the brain. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1081–1091. Association for Computational Linguistics.

Cadieu, C. F., Hong, H., Yamins, D. L., Pinto, N., Ardila, D., Solomon, E. A., Majaj, N. J., and DiCarlo, J. J. (2014). Deep neural networks rival the representation of primate IT cortex for core visual object recognition. *PLoS computational biology*, 10(12):e1003963.

Caramazza, A., Hillis, A. E., Rapp, B. C., and Romani, C. (1990). The multiple semantics hypothesis: Multiple confusions? *Cognitive neuropsychology*, 7(3):161–189.

Caramazza, A. and Shelton, J. R. (1998). Domain-specific knowledge systems in the brain: The animate-inanimate distinction. *Journal of cognitive neuroscience*, 10(1):1–34.

Caramazza, A. and Zurif, E. B. (1976). Dissociation of algorithmic and heuristic processes in language comprehension: Evidence from aphasia. *Brain and language*, 3(4):572–582.

Caucheteux, C. and King, J.-R. (2021). The Mapping of Deep Language Models on Brain Responses Primarily Depends on their Performance. working paper or preprint.

Chan, S. C., Dasgupta, I., Kim, J., Kumaran, D., Lampinen, A. K., and Hill, F. (2022). Transformers generalize differently from information stored in context vs in weights. *arXiv preprint arXiv:2210.05675*.

Chan, W., Ke, N. R., and Lane, I. (2015). Transferring knowledge from a rnn to a dnn. *arXiv preprint arXiv:1504.01483*.

Chefer, H., Gur, S., and Wolf, L. (2021). Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 782–791.

Chehab, O., Defossez, A., Loiseau, J.-C., Gramfort, A., and King, J.-R. (2021). Deep recurrent encoder: A scalable end-to-end network to model brain signals. *arXiv preprint arXiv:2103.02339*.

Chen, H. and Ji, Y. (2019). Improving the interpretability of neural sentiment classifiers via data augmentation. *arXiv preprint arXiv:1909.04225*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (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.

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

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Pre-training transformers as energy-based cloze models. In *EMNLP*.

Coenen, A., Reif, E., Yuan, A., Kim, B., Pearce, A., Viégas, F., and Wattenberg, M. (2019). Visualizing and measuring the geometry of bert. *arXiv preprint arXiv:1906.02715*.

Cohen, N., Sharir, O., and Shashua, A. (2016). On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728.

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

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Craven, M. W. (1996). *Extracting Comprehensible Models from Trained Neural Networks*. PhD thesis, University of Wisconsin-Madison.

Craven, M. W. and Shavlik, J. W. (1995). Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems 8*, NeurIPS'95, Cambridge, MA, USA. MIT Press.

Culbertson, J., Smolensky, P., and Legendre, G. (2012). Learning biases predict a word order universal. *Cognition*, 122(3):306–329.

Damasio, H., Grabowski, T. J., Tranel, D., Hichwa, R. D., and Damasio, A. R. (1996). A neural basis for lexical retrieval. *Nature*, 380(6574):499–505.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., et al. (2020). Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*.

Deacon, T. (1997). The co-evolution of language and the brain. *WW Norlon, Nueva Ymk*.

DeGroot, M. H. and Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal transformers. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR'19.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (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 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Devlin, J. T., Jamison, H. L., Matthews, P. M., and Gonnerman, L. M. (2004). Morphology and the internal structure of words. *Proceedings of the National Academy of Sciences*, 101(41):14984–14988.

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., and Smith, N. (2020). Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv: 2002.06305*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. (2021). Glam: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905*.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Ethayarajh, K. and Jurafsky, D. (2021). Attention flows are shapley value explanations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 49–54, Online. Association for Computational Linguistics.

Ettinger, A., Linzen, T., and Marantz, A. (2014). The role of morphology in phoneme prediction: Evidence from meg. *Brain and Language*, 129:14–23.

Faruqui, M. and Dyer, C. (2015). Non-distributional word vector representations. In *Proceedings of ACL*.

Frank, M. C. and Tenenbaum, J. B. (2011). Three ideal observer models for rule learning in simple languages. *Cognition*, 120(3):360–371.

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR'19.

Freitag, M., Al-Onaizan, Y., and Sankaran, B. (2017). Ensemble distillation for neural machine translation. *CoRR*, abs/1702.01802.

Friederici, A. D. (2002). Towards a neural basis of auditory sentence processing. *Trends in Cognitive Sciences*, 6(2):78–84.

Frosst, N. and Hinton, G. E. (2017). Distilling a neural network into a soft decision tree. In *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence*.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. (2018). Born-again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, ICML'18.

Fyshe, A., Talukdar, P. P., Murphy, B., and Mitchell, T. M. (2014). Interpretable semantic vectors from a joint model of brain-and text-based meaning. In *Proceedings of the conference. Association for Computational Linguistics. Meeting (ACL 2014)*, volume 2014, page 489. NIH Public Access.

Gauthier, J. and Ivanova, A. (2018). Does the brain represent words? an evaluation of brain decoding studies of language understanding. *arXiv preprint arXiv:1806.00591*.

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

Geras, K., Mohamed, A., Caruana, R., Urban, G., Wang, S., Aslan, O., Philipose, M., Richardson, M., and Sutton, C. (2016). Blending lstms into cnns. In *Workshop track, ICLR 2016*.

Giulianelli, M., Harding, J., Mohnert, F., Hupkes, D., and Zuidema, W. (2018). Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *1st BlackBoxNLP workshop at Conference on Empirical Methods in Natural Language Processing (EMNLP 2019)*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. *http://www.deeplearningbook.org*.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

Greenberg, J. (1963). Some universals of grammar with particular reference to the order of meaningful elements. *In J. Greenberg, ed., Universals of Language. 73-113. Cambridge, MA.*

Grimsley, C., Mayfield, E., and R.S. Bursten, J. (2020). Why attention is not explanation: Surgical intervention and causal reasoning about neural models. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1780–1790, Marseille, France. European Language Resources Association.

Güçlü, U. and van Gerven, M. A. J. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27):10005–10014.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17.

Hagoort, P. (2005). On broca, brain, and binding: a new framework. *Trends in Cognitive Sciences*, 9(9):416–423.

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

Hao, J., Wang, X., Yang, B., Wang, L., Zhang, J., and Tu, Z. (2019). Modeling recurrence for transformer. *arXiv preprint arXiv:1904.03092*.

Hausser, R. and Hausser, R. (2001). *Foundations of computational linguistics*. Springer.

Heilbron, M., Armeni, K., Schoffelen, J.-M., Hagoort, P., and de Lange, F. P. (2021). A hierarchy of linguistic predictions during natural language comprehension. *bioRxiv*, pages 2020–12.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hudson Kam, C. L. and Newport, E. L. (2005). Regularizing unpredictable variation: The roles of adult and child learners in language formation and change. *Language learning and development*, 1(2):151–195.

Hupkes, D., Veldhoen, S., and Zuidema, W. (2018). Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Huth, A. G., de Heer, W. A., Griffiths, T. L., Theunissen, F. E., and Gallant, J. L. (2016). Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453.

Huth, A. G., Nishimoto, S., Vu, A. T., and Gallant, J. L. (2012). A continuous semantic space describes the representation of thousands of object and action categories across the human brain. *Neuron*, 76(6):1210–1224.

Jain, S. and Huth, A. G. (2018). Incorporating context into language encoding models for fMRI. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 6629–6638, USA. Curran Associates Inc.

Jain, S. and Wallace, B. C. (2019). Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Jordan, M. I. (1997). Chapter 25 - serial order: A parallel distributed processing approach. In Donahoe, J. W. and Packard Dorsel, V., editors, *Neural-Network Models of Cognition*, volume 121 of *Advances in Psychology*, pages 471–495. North-Holland.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *CoRR*.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Khaligh-Razavi, S.-M. and Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915.

Khandelwal, U., He, H., Qi, P., and Jurafsky, D. (2018). Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Kim, C., Johnson, N. F., and Gold, B. T. (2012). Common and distinct neural mechanisms of attentional switching and response conflict. *Brain Research*, 1469:92–102.

Kim, N., Patel, R., Poliak, A., Wang, A., Xia, P., McCoy, R. T., Tenney, I., Ross, A., Linzen, T., Durme, B. V., Bowman, S. R., and Pavlick, E. (2019). Probing what different nlp tasks teach machines about function word comprehension. In **SEMEVAL*.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1885–1894. JMLR.org.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3519–3529. PMLR.

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4.

Kuhn, H. W. and Tucker, A. W. (1953). *Contributions to the Theory of Games*. Number 28. Princeton University Press.

Kuncoro, A., Dyer, C., Rimell, L., Clark, S., and Blunsom, P. (2019). Scalable syntax-aware language models using knowledge distillation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3472–3484, Florence, Italy. Association for Computational Linguistics.

Kuncoro, A., Kong, L., Fried, D., Yogatama, D., Rimell, L., Dyer, C., and Blunsom, P. (2020). Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.

Laakso, A. and Cottrell, G. (2000). Content and cluster analysis: assessing representational similarity in neural systems. *Philosophical psychology*, 13(1):47–76.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Landau, B., Smith, L. B., and Jones, S. S. (1988). The importance of shape in early lexical learning. *Cognitive development*, 3(3):299–321.

Le, P. and Zuidema, W. (2014). The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 729–739, Doha, Qatar. Association for Computational Linguistics.

LeCun, Y. and Bengio, Y. (1998). *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA.

Lee, J., Shin, J.-H., and Kim, J.-S. (2017). Interactive visualization and manipulation of attention-based neural machine translation. In *proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126.

Leffel, T., Lauter, M., Westerlund, M., and Pylkkänen, L. (2014). Restrictive vs. non-restrictive composition: a magnetoencephalography study. *Language, cognition and neuroscience*, 29(10):1191–1204.

Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4(0).

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2022). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* Just Accepted.

Liu, X., He, P., Chen, W., and Gao, J. (2019a). Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *ArXiv*, abs/1904.09482.

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

Loshchilov, I. and Hutter, F. (2017). Sgdr: Stochastic gradient descent with warm restarts. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR'17.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Luo, S., Wang, X., Fang, G., Hu, Y., Tao, D., and Song, M. (2019). Knowledge amalgamation from heterogeneous networks by common feature learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, IJCAI'19.

Maheswaranathan, N. and Sussillo, D. (2020). How recurrent networks implement contextual processing in sentiment analysis. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6608–6619. PMLR.

Maheswaranathan, N., Williams, A., Golub, M., Ganguli, S., and Sussillo, D. (2019). Universality and individuality in neural dynamics across large populations of recurrent networks. In *Advances in neural information processing systems 32*, NeurIPS'19.

Marcus, G. F., Pinker, S., Ullman, M., Hollander, M., Rosen, T. J., Xu, F., and Clahsen, H. (1992). Overregularization in language acquisition. *Monographs of the society for research in child development*, pages i–178.

Markman, E. M. (1990). Constraints children place on word meanings. *Cognitive science*, 14(1):57–77.

Markman, E. M. (1991). 3. the whole-object, taxonomic, and mutual exclusivity assumptions as initial constraints. *Perspectives on language and thought: Interrelations in development*, page 72.

Marvin, R. and Linzen, T. (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.

McCoy, R. T., Frank, R., and Linzen, T. (2020). Does syntax need to grow on trees? sources of hierarchical inductive bias in sequence-to-sequence networks. *CoRR*, abs/2001.03632.

McCoy, T., Pavlick, E., and Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Merriman, W. E., Bowman, L. L., and MacWhinney, B. (1989). The mutual exclusivity bias in children's word learning. *Monographs of the Society for Research in Child Development*, 54(3/4):i–129.

Metzger, N., Hahn, C., Siber, J., Schmitt, F., and Finkbeiner, B. (2022). Attention flows for general transformers. *arXiv preprint arXiv:2205.15389*.

Micheli, V., d'Hoffschmidt, M., and Fleuret, F. (2020). On the importance of pre-training data volume for compact language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7853–7858, Online. Association for Computational Linguistics.

Mijnheer, M. (2017). Combining experiential and distributional semantic data to predict neural activity patterns. unpublished Bachelor thesis, Artificial Intelligence, University of Amsterdam.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Miller, G. A. (1995). WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical report, Rutgers University, New Brunswick, NJ.

Mitchell, T. M., Shinkareva, S. V., Carlson, A., Chang, K.-M., Malave, V. L., Mason, R. A., and Just, M. A. (2008). Predicting human brain activity associated with the meanings of nouns. *science*, 320(5880):1191–1195.

Mobahi, H., Farajtabar, M., and Bartlett, P. L. (2020). Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*.

Morcos, A., Raghu, M., and Bengio, S. (2018). Insights on representational similarity in neural networks with canonical correlation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5732–5741. Curran Associates, Inc.

Mu, N. and Gilmer, J. (2019). Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*.

Mueller, R. R., Kornblith, S., and Hinton, G. E. (2019). When does label smoothing help? In *Advances in Neural Information Processing Systems 32*, NeurIPS'19.

Murphy, B., Talukdar, P., and Mitchell, T. (2012). Selecting corpus-semantic models for neurolinguistic decoding. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, SemEval '12, pages 114–123, Stroudsburg, PA, USA. Association for Computational Linguistics.

Murphy, B., Wehbe, L., and Fyshe, A. (2018). Decoding language from the brain. *Language, cognition, and computational models*, pages 53–80.

Neyshabur, B., Sedghi, H., and Zhang, C. (2020). What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*.

Park, W., Kim, D., Lu, Y., and Cho, M. (2019). Relational knowledge distillation. In *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR'19.

Patel, A. D. (2003). Language, music, syntax and the brain. *Nature neuroscience*, 6(7):674–681.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe:global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Pereira, F., Gershman, S., Ritter, S., and Botvinick, M. (2016). A comparative evaluation of off-the-shelf distributed semantic representations for modelling behavioural data. *Cognitive neuropsychology*, 33(3-4):175–190.

Pereira, F., Lou, B., Pritchett, B., Ritter, S., Gershman, S. J., Kanwisher, N., Botvinick, M., and Fedorenko, E. (2018). Toward a universal decoder of linguistic meaning from brain activation. *Nature communications*, 9(1):963.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (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)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2018*, pages 2227–2237.

Pezeshkpour, P., Jain, S., Wallace, B., and Singh, S. (2021). An empirical comparison of instance attribution methods for NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 967–975, Online. Association for Computational Linguistics.

Phuong, M. and Lampert, C. (2019). Towards understanding knowledge distillation. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of Machine Learning Research*, volume 97, pages 5142–5151, Long Beach, California, USA. PMLR.

Plunkett, K. and Marchman, V. (1991). U-shaped learning and frequency effects in a multi-layered perception: Implications for child language acquisition. *Cognition*, 38(1):43–102.

Pruthi, D., Gupta, M., Dhingra, B., Neubig, G., and Lipton, Z. C. (2019). Learning to deceive with attention-based explanations. *arXiv preprint arXiv:1909.07913*.

Qian, P., Qiu, X., and Huang, X. (2016). Bridging lstm architecture and the neural dynamics during reading. In *Proceedings of International Joint Conferences on Artificial Intelligence Organization (IJCAI 2016)*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *ICML*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

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

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017). Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6076–6085. Curran Associates, Inc.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR.

Riddoch, M. J., Humphreys, G. W., Coltheart, M., and Funnell, E. (1988). Semantic systems or system? neuropsychological evidence re-examined. *Cognitive Neuropsychology*, 5(1):3–25.

Rikhye, R. V., Gilra, A., and Halassa, M. M. (2018). Thalamic regulation of switching between cortical representations enables cognitive flexibility. *Nature neuroscience*, 21(12):1753–1763.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kocisky, T., and Blunsom, P. (2016). Reasoning about entailment with neural attention. In *International Conference on Learning Representations (ICLR)*.

Rowling, J. K. (1998). *Harry Potter And the Sorcerer's Stone*. Arthur A. Levine Books.

Ruan, Y.-P., Ling, Z.-H., and Hu, Y. (2016). Exploring semantic representation in brain activity using word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 669–679.

Rumelhart, D. E. and McClelland, J. L. (1986). *On Learning the Past Tenses of English Verbs*, page 216–271. MIT Press, Cambridge, MA, USA.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*.

Salle, A., Idiart, M., and Villavicencio, A. (2016a). Enhancing the lexvec distributed word representation model using positional contexts and external memory. *arXiv preprint arXiv:1606.01283*.

Salle, A., Idiart, M., and Villavicencio, A. (2016b). Matrix factorization using window sampling and negative sampling for improved word representations. *arXiv preprint arXiv:1606.00819*.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. (2018). On the information bottleneck theory of deep learning. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR'18.

Schrimpf, M., Blank, I., Tuckute, G., Kauf, C., Hosseini, E. A., Kanwisher, N., Tenenbaum, J., and Fedorenko, E. (2020). The neural architecture of language: Integrative reverse-engineering converges on a model for predictive processing. *BioRxiv*.

Schwartz, D. and Mitchell, T. (2019). Understanding language-elicited eeg data by predicting it from a fine-tuned language model. *arXiv preprint arXiv:1904.01548*.

Scott, W. A. (1962). Cognitive complexity and cognitive flexibility. *Sociometry*, 25(4):405–414.

Serrano, S. and Smith, N. A. (2019). Is attention interpretable? In *proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Seung, H. S., Sompolinsky, H., and Tishby, N. (1991). Learning curves in large neural networks. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, COLT '91.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Sigman, M. and Dehaene, S. (2008). Brain mechanisms of serial and parallel processing during dual-task performance. *Journal of Neuroscience*, 28(30):7585–7598.

Sinclair, A., Jumelet, J., Zuidema, W., and Fernández, R. (2022). Structural Persistence in Language Models: Priming as a Window into Abstract Language Representations. *Transactions of the Association for Computational Linguistics*, 10:1031–1050.

Singleton, J. L. and Newport, E. L. (2004). When learners surpass their models: The acquisition of american sign language from inconsistent input. *Cognitive psychology*, 49(4):370–407.

Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V., et al. (2022). Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Socher, R., Manning, C. D., and Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *In Proceedings of the NeurIPS'10 Deep Learning and Unsupervised Feature Learning Workshop*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (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.

Søgaard, A. (2016). Evaluating word embeddings with fMRI and eye-tracking. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 116–121. Association for Computational Linguistics.

Solomyak, O. and Marantz, A. (2010). Evidence for early morphological decomposition in visual word recognition. *Journal of Cognitive Neuroscience*, 22(9):2042–2057.

Srinivas, S. and Babu, R. V. (2015). Data-free parameter pruning for deep neural networks. In *Proceedings of the 26th British Machine Vision Conference*, BMVC'15.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. (2022). Beyond the imitation game:

Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Sudre, G., Pomerleau, D., Palatucci, M., Wehbe, L., Fyshe, A., Salmelin, R., and Mitchell, T. (2012). Tracking neural coding of perceptual and semantic features of concrete nouns. *NeuroImage*, 62(1):451–463.

Sun, J., Wang, S., Zhang, J., and Zong, C. (2019a). Towards sentence-level brain decoding with distributed representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7047–7054.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. (2019b). Patient knowledge distillation for bert model compression. In *EMNLP/IJCNLP*.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, ICML'13.

Tan, X., Ren, Y., He, D., Qin, T., Zhao, Z., and Liu, T.-Y. (2019). Multilingual neural machine translation with knowledge distillation. *arXiv preprint arXiv:1902.10461*.

Tang, J., Shivanna, R., Zhao, Z., Lin, D., Singh, A., Chi, E. H., and Jain, S. (2020). Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*.

Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O., and Lin, J. (2019). Distilling task-specific knowledge from bert into simple neural networks. *ArXiv*, abs/1903.12136.

Tenney, I., Das, D., and Pavlick, E. (2019a). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., and Pavlick, E. (2019b). What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., et al. (2022). Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Tomasello, M. (2009). *Constructing a language*. Harvard university press.

Toneva, M. and Wehbe, L. (2019). Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). *arXiv preprint arXiv:1905.11833*.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2020). Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*.

Tran, K., Bisazza, A., and Monz, C. (2018). The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736, Brussels, Belgium. Association for Computational Linguistics.

Tung, F. and Mori, G. (2019). Similarity-preserving knowledge distillation. *ArXiv*, abs/1907.09682.

Ullman, M. T., Corkin, S., Coppola, M., Hickok, G., Growdon, J. H., Koroshetz, W. J., and Pinker, S. (1997). A neural dissociation within language: Evidence that the mental dictionary is part of declarative memory, and that grammatical rules are processed by the procedural system. *Journal of cognitive neuroscience*, 9(2):266–276.

van Schijndel, M., Mueller, A., and Linzen, T. (2019). Quantity doesn't buy quality syntax with neural language models. In *EMNLP/IJCNLP (1)*, pages 5830–5836.

Vashishth, S., Upadhyay, S., Tomar, G. S., and Faruqui, M. (2019). Attention interpretability across nlp tasks. *arXiv preprint arXiv:1909.11218*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Veldhoen, S., Hupkes, D., and Zuidema, W. H. (2016). Diagnostic classifiers revealing how neural networks process hierarchical structure. In *CoCo@NIPS*.

Vig, J. (2019). Visualizing attention in transformer-based language models. *arXiv preprint arXiv:1904.02679*.

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

Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1058–1066.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Wang, T. and Cho, K. (2016). Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Wang, Y., Huang, M., Zhao, L., et al. (2016). Attention-based lstm for aspect-level sentiment classification. In *proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Wang, Y., Sereno, J. A., Jongman, A., and Hirsch, J. (2003). fmri evidence for cortical modification during learning of mandarin lexical tone. *Journal of cognitive neuroscience*, 15(7):1019–1027.

Warrington, E. K. and Shallice, T. (1984). Category specific semantic impairments. *Brain*, 107(3):829–853.

Waxman, S. R. and Kosowski, T. D. (1990). Nouns mark category relations: Toddlers' and preschoolers' word-learning biases. *Child development*, 61(5):1461–1473.

Wehbe, L., Murphy, B., Talukdar, P., Fyshe, A., Ramdas, A., and Mitchell, T. (2014a). Simultaneously uncovering the patterns of brain regions involved in different story reading subprocesses. *in press*.

Wehbe, L., Vaswani, A., Knight, K., and Mitchell, T. M. (2014b). Aligning context-based statistical models of language with brain activity during reading. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP 2014)*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Westerlund, M. and Pylkkänen, L. (2014). The role of the left anterior temporal lobe in semantic composition vs. semantic memory. *Neuropsychologia*, 57:59–70.

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

Wintner, S. (2010). Computational models of language acquisition. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 86–99. Springer.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Xu, H., Murphy, B., and Fyshe, A. (2016). Brainbench: A brain-image test suite for distributional semantic models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2017–2021.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *proceedings of International Conference on Machine Learning*, pages 2048–2057.

Yamins, D. L. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356.

Yuan, L., Tay, F. E., Li, G., Wang, T., and Feng, J. (2019). Revisit knowledge distillation: a teacher-free framework. *arXiv preprint arXiv:1909.11723*.

Zhuang, C., Yan, S., Nayebi, A., Schrimpf, M., Frank, M. C., DiCarlo, J. J., and Yamins, D. L. K. (2021). Unsupervised neural network models of the ventral visual stream. *Proceedings of the National Academy of Sciences*, 118(3):e2014196118.

Summary

Inductive Biases for Learning Natural Language

A classic question in the study of human cognition is: what are the learning biases that make it possible for them to learn and process language? A similar question can now be asked in the study of machine intelligence: to build machine learning models for language, what are the necessary inductive biases that enable learning in an efficient and generalisable manner? We need to identify the learning biases that enable the learning of natural language, and find ways to incorporate them into machine learning models.

Taking a step toward this goal, this thesis explores different techniques to illustrate the impact of inductive biases on different aspects of the solutions these models converge to. We study the sensitivity of the representational spaces of the models to different factors. Furthermore, we propose new techniques to study the attention patterns in models with attention mechanisms.

Using these techniques we study the effect of context, context length, architectural factors, and training objective on the solutions learned by different types of neural language models. We find that different choices in designing neural networks lead towards solutions with different characteristics. While some factors such as training objective and connectivity patterns lead to more divergent solutions, the final solutions are sometimes less sensitive to other factors such as scaling model size.

We build on top of prior work to study the connection between the inductive biases of language models and the underlying mechanisms in the human brain. We find that, among existing neural network architectures, recurrence has a significant role in facilitating learning structures needed to learn language more similar to the human brain. Looking further into the impact of recurrence, we identify and empirically evaluate different sources of inductive biases in recurrent neural networks: (1) sequentiality, (2) memory bottleneck, and (3) parameter sharing in time.

We demonstrate that the process of distilling knowledge from one model to another can shed light on the difference in the inductive biases and expressivity of the teacher and student model. Moreover, we find that some of the effects of inductive biases can potentially transfer through knowledge distillation.

In the end, considering the recent impressive progress in deep learning, and the contribution of the scaling factor in this progress, we believe it is important to have evaluation frameworks that allow us to understand the different ways in which models generalize in different settings and under different conditions. In this thesis, we take a small step toward building such frameworks.

Samenvatting

Inductieve Biases voor het Leren van Natuurlijke Taal

Een klassieke vraag in het onderzoek naar menselijke cognitie is: welke voorkeuren en eigenaardigheden van het menselijk brein maken het mogelijk om taal te leren en te gebruiken? Een vergelijkbare vraag kunnen we tegenwoordig stellen over machine-intelligentie: hoe ontwerpen we leeralgoritmes voor natuurlijke taal die efficiënt zijn en die computers in staat stellen de juiste generalisaties te maken? Wat zijn de noodzakelijke 'learning biases' en hoe bouwen we ze in in computermodellen?

In dit proefschrift onderzoeken we technieken om de impact van verschillende biases in kaart te brengen. We bestuderen de representaties van taal die de modellen leren, en hoe afhankelijk ze zijn van verschillende factoren. In het bijzonder kijken we naar modellen die gebaseerd zijn op het mechanisme van 'multi-head attention', en werken we technieken uit om de patronen in dit mechanisme inzichtelijk te maken.

Met behulp van die technieken onderzoeken we verschillende neurale taalmodellen en de effecten die verschillen in context en context-lengte, architectuur en 'training objective' hebben op de oplossingen die die modellen leren. Sommige van deze factoren, zoals training objective en connectiviteit in de architectuur maken veel verschil. Andere factoren, zoals het aantal lagen in de architectuur, hebben minder effect op de uiteindelijke oplosisngen.

We bouwen voort op eerdere onderzoeken om de relatie tussen de biases van taalmodellen en de onderliggende neurale mechanismen in de menselijke hersenen te bestuderen. In de bestaande neurale netwerkarchitecturen blijkt vooral 'recurrentie' een significante rol te spelen bij het leren van structuren die nodig zijn om taaltaken op te lossen die meer lijken op die van de menselijke hersenen. Verder identificeren en evalueren we empirische bronnen van biases in recurrente neurale netwerken: (1) sequentialiteit, (2) geheugenbottleneck en (3) het delen van parameters over tijd.

We laten zien dat het proces van het destilleren van kennis van het ene model naar het andere inzicht kan geven in de verschillen in de inductieve biases en de expressiviteit van het docent- en studentenmodel. Bovendien ontdekken we dat sommige effecten van de 'learning biases' mogelijk worden overgedragen via kennisdestillatie.

Dit proefschrift vormt dus een bijdrage aan de analyse van deep learning modellen van taal. Gezien de enorme, recente vooruitgang op dit gebied, en de cruciale rol die schaalvergroting daarbij heeft gespeeld, is een goede evaluatie van taalmodellen enorm belangrijk. Evaluatie-technieken moeten ons in staat stellen om de verschillende manieren te begrijpen waarop modellen generaliseren, met verschillende parameters en onder verschillende omstandigheden. In dit proefschrift zetten we een kleine stap naar het bouwen van een raamwerk voor dergelijke evaluaties.

Titles in the ILLC Dissertation Series:

ILLC DS-2018-03: **Corina Koolen**
Reading beyond the female: The relationship between perception of author gender and literary quality

ILLC DS-2018-04: **Jelle Bruineberg**
Anticipating Affordances: Intentionality in self-organizing brain-body-environment systems

ILLC DS-2018-05: **Joachim Daiber**
Typologically Robust Statistical Machine Translation: Understanding and Exploiting Differences and Similarities Between Languages in Machine Translation

ILLC DS-2018-06: **Thomas Brochhagen**
Signaling under Uncertainty

ILLC DS-2018-07: **Julian Schlöder**
Assertion and Rejection

ILLC DS-2018-08: **Srinivasan Arunachalam**
Quantum Algorithms and Learning Theory

ILLC DS-2018-09: **Hugo de Holanda Cunha Nobrega**
Games for functions: Baire classes, Weihrauch degrees, transfinite computations, and ranks

ILLC DS-2018-10: **Chenwei Shi**
Reason to Believe

ILLC DS-2018-11: **Malvin Gattinger**
New Directions in Model Checking Dynamic Epistemic Logic

ILLC DS-2018-12: **Julia Ilin**
Filtration Revisited: Lattices of Stable Non-Classical Logics

ILLC DS-2018-13: **Jeroen Zuiddam**
Algebraic complexity, asymptotic spectra and entanglement polytopes

ILLC DS-2019-01: **Carlos Vaquero**
What Makes A Performer Unique? Idiosyncrasies and commonalities in expressive music performance

ILLC DS-2019-02: **Jort Bergfeld**
Quantum logics for expressing and proving the correctness of quantum programs

ILLC DS-2019-03: **András Gilyén**
Quantum Singular Value Transformation & Its Algorithmic Applications

ILLC DS-2019-04: **Lorenzo Galeotti**
The theory of the generalised real numbers and other topics in logic

ILLC DS-2019-05: **Nadine Theiler**
Taking a unified perspective: Resolutions and highlighting in the semantics of attitudes and particles

ILLC DS-2019-06: **Peter T.S. van der Gulik**
Considerations in Evolutionary Biochemistry

ILLC DS-2019-07: **Frederik Möllerström Lauridsen**
Cuts and Completions: Algebraic aspects of structural proof theory

ILLC DS-2020-01: **Mostafa Dehghani**
Learning with Imperfect Supervision for Language Understanding

ILLC DS-2020-02: **Koen Groenland**
Quantum protocols for few-qubit devices

ILLC DS-2020-03: **Jouke Witteveen**
Parameterized Analysis of Complexity

ILLC DS-2020-04: **Joran van Apeldoorn**
A Quantum View on Convex Optimization

ILLC DS-2020-05: **Tom Bannink**
Quantum and stochastic processes

ILLC DS-2020-06: **Dieuwke Hupkes**
Hierarchy and interpretability in neural models of language processing

ILLC DS-2020-07: **Ana Lucia Vargas Sandoval**
On the Path to the Truth: Logical & Computational Aspects of Learning

ILLC DS-2020-08: **Philip Schulz**
Latent Variable Models for Machine Translation and How to Learn Them

ILLC DS-2020-09: **Jasmijn Bastings**
A Tale of Two Sequences: Interpretable and Linguistically-Informed Deep Learning for Natural Language Processing

ILLC DS-2020-10: **Arnold Kochari**
Perceiving and communicating magnitudes: Behavioral and electrophysiological studies

ILLC DS-2020-11: **Marco Del Tredici**
Linguistic Variation in Online Communities: A Computational Perspective

ILLC DS-2020-12: **Bastiaan van der Weij**
Experienced listeners: Modeling the influence of long-term musical exposure on rhythm perception

ILLC DS-2020-13: **Thom van Gessel**
Questions in Context

ILLC DS-2020-14: **Gianluca Grilletti**
Questions & Quantification: A study of first order inquisitive logic

ILLC DS-2020-15: **Tom Schoonen**
Tales of Similarity and Imagination. A modest epistemology of possibility

ILLC DS-2020-16: **Ilaria Canavotto**
Where Responsibility Takes You: Logics of Agency, Counterfactuals and Norms

ILLC DS-2020-17: **Francesca Zaffora Blando**
Patterns and Probabilities: A Study in Algorithmic Randomness and Computable Learning

ILLC DS-2021-01: **Yfke Dulek**
Delegated and Distributed Quantum Computation

ILLC DS-2021-02: **Elbert J. Booij**
The Things Before Us: On What it Is to Be an Object

ILLC DS-2021-03: **Seyyed Hadi Hashemi**
Modeling Users Interacting with Smart Devices

ILLC DS-2021-04: **Sophie Arnoult**
Adjunction in Hierarchical Phrase-Based Translation

ILLC DS-2021-05: **Cian Guilfoyle Chartier**
A Pragmatic Defense of Logical Pluralism

ILLC DS-2021-06: **Zoi Terzopoulou**
Collective Decisions with Incomplete Individual Opinions

ILLC DS-2021-07: **Anthia Solaki**
Logical Models for Bounded Reasoners

ILLC DS-2021-08: **Michael Sejr Schlichtkrull**
Incorporating Structure into Neural Models for Language Processing

ILLC DS-2021-09: **Taichi Uemura**
Abstract and Concrete Type Theories

ILLC DS-2021-10: **Levin Hornischer**
Dynamical Systems via Domains: Toward a Unified Foundation of Symbolic and Non-symbolic Computation

ILLC DS-2021-11: **Sirin Botan**
Strategyproof Social Choice for Restricted Domains

ILLC DS-2021-12: **Michael Cohen**
Dynamic Introspection

ILLC DS-2021-13: **Dazhu Li**
Formal Threads in the Social Fabric: Studies in the Logical Dynamics of Multi-Agent Interaction

ILLC DS-2022-01: **Anna Bellomo**
Sums, Numbers and Infinity: Collections in Bolzano's Mathematics and Philosophy

ILLC DS-2022-02: **Jan Czajkowski**
Post-Quantum Security of Hash Functions

ILLC DS-2022-03: **Sonia Ramotowska**
Quantifying quantifier representations: Experimental studies, computational modeling, and individual differences

ILLC DS-2022-04: **Ruben Brokkelkamp**
How Close Does It Get?: From Near-Optimal Network Algorithms to Suboptimal Equilibrium Outcomes

ILLC DS-2022-05: **Lwenn Bussière-Carae**
No means No! Speech Acts in Conflict

ILLC DS-2023-01: **Subhasree Patro**
Quantum Fine-Grained Complexity

ILLC DS-2023-02: **Arjan Cornelissen**
Quantum multivariate estimation and span program algorithms

ILLC DS-2023-03: **Robert Paßmann**
Logical Structure of Constructive Set Theories

ILLC DS-2023-04: **Samira Abnar**
Inductive Biases for Learning Natural Language