Syntactic Knowledge Distillation
Between an RNNG and a Transformer

by

JACK SETTLES

(Under the direction of John Hale)

ABSTRACT

Language modeling casts a probability distribution over words or sub-word tokens given an input sequence of the same kind. This distribution represents the likelihood that each of those elements occurs next in the input sequence. Modern language models rely on vast amounts of training data and compute to achieve human like natural language generation and understanding. The major flaw in this approach is that these models treat sentences as purely linear combinations of words - beads on a string. Language is notoriously hierarchical, though. This means that only certain words can occur in certain locations based on the syntactic structure up to that point. This work aims to leverage syntactic knowledge in language modeling by distilling the knowledge from a cumbersome syntactic language model into a more optimized sequential language model.

Index words: Computational Linguistics, language modeling, knowledge distillation, formalist linguistics, functionalism, RNNGs, cognitive plausibility

Syntactic Knowledge Distillation Between an RNNG and a Transformer

by

JACK SETTLES

A.B., University of Georgia, 2021

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2025

Syntactic Knowledge Distillation Between an RNNG and a Transformer

by

JACK SETTLES

Approved:

Major Professor: John Hale

Committee: Geng Yuan
Vera Lee-Schoenfeld

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
May 2025

ACKNOWLEDGMENTS

I would like to thank a number of people who contributed to this work at some point or another along the way. I would like to thank O.B. Bassler for instilling in me a deep love for language, and all that it entails. Without him, I would probably be doing something very different at the moment. I would like to thank Donald Dunagan for his unwavering aid as a T.A. in NLP, and as a resource for semesters after that, especially with helping me navigate the UGA research and Corpus clusters, and graduate school in general. I would like to thank Zhuofei Hou for his technical support as I encountered errors with the computing cluster environments. I would like to thank Vera Lee-Schoenfeld for her thorough and exhaustive efforts as a teacher of syntax, especially to someone who comes from outside formalist linguistics. I would like to thank Geng Yuan for his unwavering support in teaching me techniques of deep learning, and pushing me to pursue further graduate studies. Last, but not least, I would like to thank John Hale for being a phenomenal and passionate teacher, for allowing me to direct my studies as I saw them unfold, and always offering guidance along the way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

§ 1.1 Overview

Broadly speaking, the task of language modeling amounts to casting a probability distribution over a vocabulary of words, or sub-word tokens, which represents the likelihood that each of those elements occurs next given a prior sequence of words or sub-word tokens. Various different approaches to this task have come and gone over the years, spanning from statistical corpus methods such as N-gram models (Chen and Goodman, 1999), to various neural language models such as recurrent neural networks (RNNs) (Rumelhart et al., 1986; Elman, 1990), LSTMs (Hochreiter and Schmidhuber, 1997), GRUs (Cho et al., 2014), and finally Transformer (Vaswani et al., 2017) based models (Radford et al., 2018; Devlin et al., 2019). Recent advances in language modeling have resulted in extraordinary performance, both for natural language generation, as well as down-stream task performance. However, the development of these methods has been accompanied by an increase in size, both for the language models themselves, as well as the amount of data required to train them. As of the time of this writing (2024), current state of the art models require datasets that contain words in the order of trillions, far exceeding any amount a human has ever consumed by the time they acquire language. Therefore, the central aims of this work are to 1) address the data inefficiencies of modern large language models (LLMs), and 2) to evaluate whether their latent linguistic knowledge can be ascertained on a dataset with a size roughly comparable

to that of which a human experiences as they acquire language.

§ 1.2 Motivation

The motivations behind this work stem from both practical/engineering concerns as well as scientific matters. As for the practical concerns, current language models need to be trained on billions-trillions of words in order to reach desired performance levels. Desired performance, in this case, is specific to the application that the language model is being built for. As an example, many current LLMs are being built with the goal of creating a generalist/omniscient expert on a vast array of subjects. The value behind this is non-trivial: entire suites of applications can be built off language models that are both experts on language and experts on a wide variety of subjects. These applications can ultimately replace mundane and time consuming tasks that humans have had to carry out manually up until recently. LLMs also have the potential to aid in protein sequencing and drug development/discovery, displaying their potential as tools for scientific discovery in other fields. So, there is a practical concern behind training language models in a more efficient manner. They are useful, but they can be time consuming to train, and they can be both financially and environmentally costly. The more scientific concern behind current LLMs pertains to their potential use for scientific discovery in the fields of linguistics, cognitive science, and artificial intelligence (to name a few). As mentioned previously, humans only need to be exposed to less than 100M words by their early teenage years to grasp natural language (Gilkerson et al., 2017). In stark contrast, most modern LLMs require orders of magnitude more than that: BERT was trained on roughly 3.3 billion words, and GPT-3 and LLaMa 2 were trained on 500 billion and 2 trillion tokens, respectively (Devlin et al., 2019; Brown et al., 2020; Touvron et al., 2023b). Much of these sizes can be attributed to the desire to build expert-level assistants on a wide array of subjects. Regardless, when trained on a cognitively-plausible sized dataset, language models tend to underperform on

language modeling as well as on down-stream tasks. So, if the scientific community wishes to use modern language models as an approximate model of cognition in humans, the learning conditions must be the same.

§ 1.3 Contribution

The primary contributions of this work are as follows:

- Explore various techniques that reduce the amount of training data required for building language models
- Evaluate these models on a handful of benchmarks that assess a model's understanding of various linguistic phenomena

To be more specific, this paper explores some architectural modifications to decoder-only transformer style language models (like those of the GPT series from OpenAI), as well as knowledge distillation (KD) with the help of more competent, but more cumbersome, teacher models. As a teacher model, we will be using a recurrent neural network grammar (RNNG) (Dyer et al., 2016). RNNGs are trained to model language sequentially, as well as model the syntactic structure of a sentence as it is being processed. This is done by simultaneously minimizing the negative log likelihood loss (NLL loss) associated with next word prediction, as well as minimizing the NLL loss associated with predicting the right sequence of parsing actions for that sentence based on a SHIFT-REDUCE parser. The aim of KD is to get a student model (the transformer in this case) to mimic the output of the teacher model by aligning their vocabulary probability distributions. In doing so, the hope is that the student model can learn predictions that are syntactically informed from the teacher without ever being explicitly trained to model syntax. To assess whether the student model achieves this, we will evaluate the model on the BLiMP benchmark (Warstadt et al., 2020), which contains 67,000 test cases covering various syntactic, semantic, and morphological phenomena.

§ 1.4 Thesis Structure

The rest of this paper follows as such: Chapter 2 consists of a literature review on a swath of topics, including a background on different computational approaches to language modeling, the costs they incur, their theoretical connections in linguistics, and knowledge distillation. Chapter 3 details the implementation and methods of this research. Specifically, we will cover some architectural modifications for addressing data inefficiencies found in recent literature, the knowledge distillation setup, and the evaluation pipeline. Chapter 4 presents the results, chapter 5 provides a discussion of those results, and chapter 6 concludes by acknowledging some limitations and potential future work that can be built from this project.

Chapter 2

Literature Review

§ 2.1 Data Efficiency: An Extrinsic and Intrinsic Good

The task of language modeling involves predicting the next word or token in a sequence given the previous words or tokens. Before 2017, many such models were built using recurrent neural network architectures (Rumelhart et al., 1986; Elman, 1990). Intuitively, this makes sense: language is processed sequentially, and RNNs also process its input elements sequentially. However, these models struggle to capture long-range dependencies within texts due to vanishing/exploding gradients during training. This can especially be seen when contexts get increasingly further away from the current word being processed. For example, when there are a lot of intervening words between a subject and a pronoun like *him*, these models struggle to realize that these two words are referring to the same entity (the previously stated subject).

Vaswani et al. (2017) introduced the transformer model architecture, an encoder-decoder model tasked with the sequence-to-sequence objective of language translation. Since then, transformers have been adopted and modified for the task of language modeling. In some cases, such as BERT (Devlin et al., 2019), the model is an encoder-only; however, it has become customary to use a decoder-only architecture these days. There are two features of transformers that give them an advantage over traditional RNN models. The first advantage is their powerful "self-attention" mechanism, which is an adaptation of the original attention

mechanism used in Bahdanau et al. (2014). At a high level, self-attention allows words to essentially score how correlated they are with the rest of the words in the text. The second is that these models allow all words in a sequence to be processed in parallel, rather than sequentially. This allows the self-attention mechanism to look through the entire sequence of context words, capturing dependencies between words that could span thousands of tokens. In an RNN, if one wanted to compute the attention score between a word and a previous context word, the model would be limited by how far back the context word is in relation to the current word. This is because the activation of that previous context word would eventually diminish as words are further processed through the RNN at each time step. Since transformers process all tokens at once, input sequences can be significantly larger than those used in RNNs. The result has been a dramatic increase in model sizes: while the LSTMs trained by Sutskever et al. (2014) were certainly large for the time period (about 384M parameters), the original transformer model, in its first iteration, contained roughly the same scale of parameters (about 213M) (Vaswani et al., 2017). So, the two model types sat on the same order of size with about 10^8 parameters. However, for Transformers, it only ballooned from there. A mere two years later in 2019, OpenAI's GPT-2 was released at 1.5B parameters (Radford et al., 2019), and by 2022, Google's PaLM had reached 540B parameters, 360 times the size of GPT-2 (Maslej et al., 2023).

These models further gained even more attention in November 2022 when OpenAI was the first to release the web-interface chat version of GPT-3, ChatGPT. This model, along with many others that have since been released, can perform remarkably well on a suite of tests. They can answer complex and nuanced questions, complete prompts the length of entire documents, perform zero-shot inference where no examples are needed for them to learn a behavior, and even pass standardized tests such as the LSAT, GRE, and some AP exams (Millière and Buckner, 2024). These successes have led many to believe that the question of how to accurately model human language has been adequately answered. However, just

because a machine learning model can mimic the linguistic behavior of humans does not necessarily entail any understanding of language. This was the essence of John Searle's Chinese Room thought experiment (Searle, 1980). The mere output of coherent language does not, on its own, constitute language understanding. Ideally, we could investigate the internal representations of a language model to see if they even remotely align with the internal representations of humans. Even if they do, though, we must then ask ourselves how these language models acquired their understanding of the world. If there is a significant divergence between how humans acquire language and how language models acquire language (for instance, in terms of quantity of words required to grasp language), then closing this gap becomes the next concern in order to generalize about how the human language faculty is acquired and how it functions.

So, we wish to probe the "inner world" of a language model. There is a problem with this inquiry at the outset: how we do that is not exactly obvious. If vector representations of words or hidden activation states were sufficient, then naturally we would say that language models contain world models. However, that gets us nowhere; on their own, vector representations of words are opaque—a list of 512 (or some other number of dimensions) real numbers does not jump out to a human with any kind of semantic content. Similarly, clusters of neurons firing between each other is equally opaque. So, for both humans and language models, we need to adjust our expectations for how we can investigate an internal world model. By "internal world model", we presumably mean something similar to how humans can relate words and concepts together. Do LLMs hold the same linguistic relationships that humans do? Can they infer synonyms and antonyms to words, or can they understand part-whole and type-subtype relationships? These are merely a few of the scientific questions we have with respect to a language model's knowledge (a Chomskyan might call this their "competence"). Since we cannot interpret high dimensional vectors directly, all of these evaluations must be based on the behavior of the model. So, the very thing we

wish to know—the internal representations of language—are inherently opaque to us, and outward criteria must suffice for now. Searle's Chinese Room certainly shows us the gap between language/cognition and our understanding of it. However, it leaves one thinking there is more to know about the mind than what is readily available for us. Our epistemology prevents that "more" from us. As the late American philosopher Stanley Cavell says in his magnum opus *The Claim of Reason*, when discussing the nature of an "inner process", he describes someone as "throwing out [their] signals" of it—their behavior. When another person has "had the opportunity to be apprised of [their] inner world, then *do* [they] really know it (know *it*) or do the signals come from a source [one] can never check, hence signify something [one] can never know" (Cavell, 1979, p. 97). The point in bringing up Searle in the first place was not to show that he had the wrong idea. The point was to show the very human urge to look and uncover more, even when we may have already hit a bedrock of interpretability. The point can be scientifically haunting, but also invigorating. It shows us that we must do more with what we have rather than simply search for more. We can directly interpret a model's behavior, but not its hidden states. It follows then that we must craft our investigations with more care. The tests must be more fine grained. As we will see later on, BLiMP (Warstadt et al., 2020) is a step in the right direction.

While modern LLMs have proven to be useful for a number of routine tasks, they nonetheless require enormous amounts of computing resources, money, time, and tokens to build. Take the BLOOM language model, which has a model size comparable to that of GPT-3 at 176B and 175B parameters, respectively. BLOOM took over 118 days, 384 NVIDIA A100 GPUs, and nearly 1.1 million GPU hours to train (Luccioni et al., 2022). At the time of this writing (January, 2025), a quick online search for the price of one NVIDIA A100 shows that they can cost anywhere from 8k-17k US dollars, depending on their memory capacity. So, the cost of hardware to train LLMs today is certainly a barrier to entry.

Infrastructure costs are merely up front though. To actually carry out the training pro-

cess, researchers also need to be able to afford the power bill, which can be quite enormous given that these computing clusters are running for days non-stop carrying out lots of computations. Released in 2019, the cost of training GPT-2 was 50,000 USD; just three years later, the cost to train BLOOM was at 2.3 million USD, and Google's 540B parameter PaLM model reached 8 million USD (Maslej et al., 2023). The cost of training is directly correlated with the number of parameters and FLOPs in a model. In deep learning, a FLOP is a "floating point" (number) operation, such as addition, subtraction, multiplication and division. At their core, neural networks are almost entirely FLOPs: multiplication, addition, division, etc. with matrices. This means that these operations are all happening in parallel, hence the necessity to use GPUs which can handle parallelization, unlike CPUs. While parameter counts give an estimate for model size, FLOPs provide an estimate for how many times a model has to carry out arithmetic processes. So, the more FLOPs a model has to compute, the longer it will take to train, the more power will be consumed, and the more money it will cost.

There is another cost associated with training LLMs, though, that has not been addressed: carbon. Given the amount of time and energy required to train these models, researchers have started to devote more effort towards tracking the carbon footprint produced by them. Luccioni et al. (2022) estimate that BLOOM produced nearly 25 tonnes of CO_2 equivalents (CO_2e) just during training alone, and just over 50 tonnes when accounting for other processes carried out by the computing cluster. They also estimate that GPT-3 produced roughly 500 tonnes during its training process based on the information provided in the initial paper for the model (Brown et al., 2020). For perspective, one passenger traveling by plane from New York to San Francisco and back produces about 1 tonne of CO_2e, and on average a human produces about 5.5 tonnes per year (Strubell et al., 2019). So, there are practical and environmental concerns over the mass production of transformer based LLMs.

Another concern that pertains to modern LLMs has to do with the amount of data that

is required to train them. This is certainly still a practical concern: more data means more computational power required to both store the data and to carry it all through training, plus an extensive power bill, which leads back into concerns over the carbon footprint. However, the more scientific concern associated with this much data can best be summed up in one question: why do LLMs require so much data in order to converge properly (i.e. generate grammatical text and pass various NLP benchmarks)? A human can acquire nearly full linguistic capabilities after encountering less than 100M words (Warstadt et al., 2023; Gilkerson et al., 2017). In comparison, LLMs are trained on datasets in the order of trillions of tokens. They have effectively read the entirety of the internet. This is great for when one needs to quickly ask a specific question (and hope that the model is not suffering from a "hallucination" and giving a false answer). If the goal is to build an LLM to serve as an expert assistant on a topic, then naturally an engineer will use as much data as possible. But if the question is whether LLMs are adequate cognitive models of human language, there is more work to be done.

§ 2.2 Types of Language Models

§ 2.2.1 "Vanilla" Language Models

In their most basic implementations, language models are sequential: they predict the next element of a sequence given a previous part of the sequence. Within this category of sequential models, there are two main subtypes: autoregressive and autoencoding. Autoregressive models can be implemented as either RNN or transformer architectures, and their training objective is to predict the next word or token in a sequence based on the preceding words or tokens. This prediction is then appended to the end of the input sequence, which is then used to predict the following next word or token. This is done until some sort of end-of-sequence token is generated. Some common autoregressive transformers include the GPT and Llama series models (Radford et al., 2018, 2019; Brown et al., 2020; Touvron et al.,

2023a,b). Autoencoding models, on the other hand, attempt to reconstruct an input sequence of words/tokens where some of them have been masked. They do so by producing dense vector representations of this text which is then used to predict the missing tokens. This is known as masked language modeling (MLM). Typically about 10-15% of the words in a training dataset are covered with a special MASK token during the training process. A common autoencoding language model is BERT - Bidirectional Encoder Representations from Transformers (Devlin et al., 2019). Because it is not trying to predict the next word to generate text, these models benefit from context on both sides of the target. With that being said, neither of these variations - autoregressive or autoencoding - are anything extraordinary. In fact, next word prediction is the very essence of language modeling, and BERT's training objective of MLM is very similar to the continuous bag of words implementation of Word2vec: predicting a word based on its surrounding context (Mikolov et al., 2013). In short, most language models operate on a flat structure, treating words as purely sequential elements. However, sequential language models are not the only type of language models.

There is one key problem to the approach that sequential language models take: as Dyer et al. (2016) put it, "sequential models are a priori inappropriate models of natural language, since relationships among words are largely organized in terms of latent nested structures rather than sequential surface order". Here, Dyer cites the book *Syntactic Structures* by Chomsky (1957), and Chomsky himself is considered the leading figure of modern formalist linguistics. As a result of this apparent flaw in language models, Dyer et al. (2016) introduce a new type of language model to circumvent this lack of hierarchy in "Vanilla" implementations: Recurrent Neural Network Grammars (RNNGs).

§ 2.2.2 RNNGs: Syntax Experts

At the implementation level, RNNGs are composed of multiple sub-models in one. One part is in charge of modeling language sequentially by predicting the next token in a sequence.

This is standard autoregressive language modeling. Another part of the model is in charge of modeling the parsing actions required to generate a syntax tree for the given sentence. These are based on a top-down shift-reduce parsing strategy. By top down, what is meant is that the model begins with the highest non-terminal node in a syntax tree, namely the root (S for sentence). For example, take the sentence "The markets are returning to normalcy" from the Penn Treebank (Marcus et al., 1993). Its string representation can be written using parentheses as shown below, with the accompanying syntax tree shown in Figure 2.1:

```
(S (NP-SBJ (DT The) (NNS markets)) (VP (VBP are) (VP (VBG returning) (PP-DIR
(TO to) (NP (NN normalcy))))) (.  .))
```

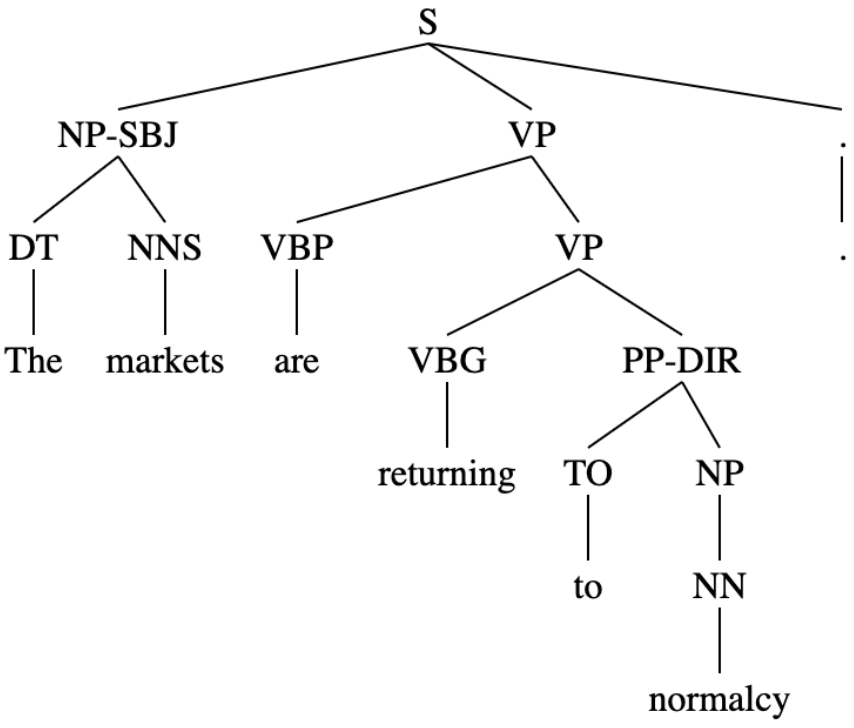

Figure 2.1: Example syntax tree

The highest level, represented by the opening parenthesis followed by 'S', contains the entire sentence. At the next level down, the noun phrase subject and verb phrase are introduced, and so on until we hit the parts of speech labels and terminal nodes (an example of each would be (NN normalcy)). Parse strings for sentences like the one above serve as the

training data for RNNGs. The sequence of words from the string is collected and converted into word/token IDs, as well as a list of parsing actions. These actions are based on the shift-reduce parser. In the RNNG implementation of this parser, there are two separate data structures that maintain the current state of the parsing process: a stack and a buffer. The stack contains partially or fully completed constituents in the sentence, while the buffer contains unprocessed elements of the sentence. There are also three actions that the parser can make: shift, reduce, or open a non-terminal '(NT'. At the beginning of the parsing process, the stack is empty, and the entire string sits in the buffer. The model then has to learn which actions to perform to model the syntactic structure of the sentence. Seeing that every sentence starts with the opening of the non-terminal node '(S', the first correct action would be to open a '(NT' and add that to the stack. This is done for all non-terminals encountered in the buffer. When the model then encounters a word or sub-word token, it SHIFTs that element onto the top of the stack. When the model encounters a closing parenthesis ')', it applies a REDUCE operation. Reduce effectively closes a constituent, so that everything between the most recently opened non-terminal '(NT' and the current closing parenthesis ')' is together. So for example, the separate elements of (DT The) and (NNS markets) are combined into (NP-SBJ (DT The) (NNS markets)) when the second parenthesis after 'markets' is processed. This selection of actions is performed for every opening '(NT', word/token, and closing parenthesis in a string until the buffer is empty, and the only thing in the stack is the entire combined sentence, all under the constituent 'S'.

By learning the latent structure of a sentence, a model can learn when to selectively generate a new word, and one that is more accordant with the current state of the parse tree for a sentence. Before venturing further into how we tie these language models together, it is worth detailing where sequential and syntactic models differ in their approach to language at a theoretical level first. The precursors to their respective paradigms can be traced back to the linguistic debate between formalism and functionalism.

§ 2.3 Formalism and Functionalism

In this section we briefly describe the theoretical similarities and differences between formalism and functionalism in linguistics, along with how they come together in computational applications like language modeling.

§ 2.3.1 Formalism

In contrast to its counterpart of functionalism (which is much more internally spread out), the internal structure of formalism in linguistics tends to all center around the work of Noam Chomsky himself (Thomas, 2020). Even those scholars who oppose Chomsky within formalism still tend to do so by locating themselves in relation to his work, especially his generative theory. Plainly stated, one of the hallmarks of formalist linguistics is attempting to understand all that is *possible* in a language. There is much more concern with linguistic *competence* (what speakers *know* about language to be true) than there is concern with linguistic *performance* (what speakers *actually* say and do with language). Thomas calls the formalist an "engineer" in the sense that they are attempting to build, from the ground up, a system of rules that can generate all possible grammatical utterances in a language while generating no ungrammatical ones. The linguistic data that is used to construct these grammatical rules tends to be "highly stylized, in the sense that [formalists engineer] it to demonstrate specific language properties rather than to depict actual speakers' actual usage" (Thomas, 2020, p. 9). In some cases, these examples can capture many trends in language, and are highly generalizable; however, in other cases, they can be highly curated and rare in everyday use. In constructing the grammar of a language, a formalist will primarily begin by considering expressions that are judged/deemed to be grammatically acceptable by most native speakers of that language. From there, rules are constructed, applied to more test cases, modified, and reapplied until nearly all forms of that particular expression have been explored. The ultimate goal is to be able to create a grammar that contains as few rules as

possible while still being able to construct any grammatical utterance in a language.

Part of the rift between formalism and functionalism centers on the number of rules the formalist may try to introduce to a grammar, along with the complexity of those rules given the evidence for them. For every new linguistic phenomenon, a rule must be able to account for it. If one does not exist, either the current ones need to be modified, or a new one must be added. But if too many rules start to fill up a grammar, a non-formalist may object saying that those "rules" are starting to apply to highly specific exceptions. Ideally one rule can be applied to a plethora of different utterances. This proliferation of rules in some sense helped steer Chomsky's work throughout the latter half of the 20th century, from Early Transformational Grammars in the 1960s, to Government and Binding Theory in the 1980s, and eventually to Minimalism in the 1990s (Thomas, 2020). With each iteration, rules became more abstract in an effort to contain the growth of the grammar.

Another important assumption that is found in formalist linguistics is that of the autonomy of syntax from semantics and discourse. Thomas references Andrea Moro's text *Impossible Languages* (2016, p. 2) when he claims that languages " 'must be endowed with other general properties that make them not only possible but *usable*' " (Thomas, 2020, p. 9) (emphasis added). Moro specifies that included in these properties is the capacity to carry meaning and communicate. With this, Moro is depicting a view of language where syntax and other language modules (namely semantics) sit at the same level within the hierarchy, seemingly negating the autonomy of syntax. However, Thomas points out that Moro then turns completely around to remark that " 'these properties are trivial' " (Thomas, 2020, p. 9). We highlight this apparent contradiction to show that this formalist/functionalist split is not clear cut, and it can run right down the middle of an individual. Nonetheless, to Moro (and some other formalists), the communicative properties of language are secondary to the rules of syntax. Syntax would, and does, exist on its own, even if the need to communicate ceased to exist. It is worth mentioning that not all formalists take this stance. It can be

seen as a spectrum where there is some internal dispersal among the views of formalism, especially as linguistics has matured.

To summarize the formalist's perspective in a way that ties back to RNNGs, syntax and the rules associated with it are essentially innate. In the case of an RNNG, these rules are explicitly encoded in the data and baked into the model's weights during training thanks to the built in parsing mechanism and objective function. There is a loose connection between this innate parsing objective in RNNGs and Chomsky's idea of a "Universal Grammar" (UG): although the Penn Treebank style annotations only represent a particular form of an English grammar, UG posits that all humans are endowed with a specific autonomous syntax module to explain how syntax can be acquired. This does not imply that all humans are born with syntactic knowledge of a specific language. That is learned through exposure to that language in a linguistic community. Rather, UG simply means that humans begin with innate universal structures ("principles and parameters") that make acquiring language and syntax even possible. So even though humans process language sequentially—one word after the other, whether reading, listening or speaking—there is a constant syntactic mechanism at work that is structuring and organizing words in a hierarchical way specified by the grammar of that language. The same can be said for the built-in parsing mechanism in an RNNG.

§ 2.3.2 Functionalism

Functionalism is less centripetal in its internal structure than its counterpart of Formalism: there is hardly one singular scholar, work, or theory dominating its camp. A great way to understand the split between formalism and functionalism lies in their respective views on the autonomy of syntax. As previously noted, formalists believe strongly in the autonomy of syntax, where it is independent from other cognitive faculties and needs. In stark contrast, the functionalist views language (syntax included) "as formed by its role in communicating meaning within a social context" (Thomas, 2020, p. 46). With this view, language cannot be

reduced to abstract formal rules that are internal to themselves. Rather, the rules took form because of some kind of external pressure. That pressure could be the need to communicate information from one person to another, and some might add the need to do that *efficiently* (Levshina, 2022). A complex super-system of systems emerges from this view.

In functionalism, there is also much less emphasis on linguistic competence, and more on linguistic performance. The functionalist does not wish to show what is grammatical and what is ungrammatical, "but rather to understand why certain speakers, in certain contexts, with certain intentions, encode messages the way they do" (Thomas, 2020, p. 47). There is a much stronger emphasis on looking at what speakers actually say, so examples tend to come from transcribed conversations and interactions. This is reminiscent of Ludwig Wittgenstein's urge to "look and see" in his *Philosophical Investigations* (1953, § 66; PI from here on). In *This New Yet Unapproachable America* (1989, p. 34; NYUA from here on), American philosopher Stanley Cavell describes Wittgenstein's itinerary in PI by referring to section §116, where Wittgenstein asks himself "is the word ever actually used in this way in the language in which it is at home?". He further describes his agenda in that section as attempting "to bring words back from their metaphysical to their everyday use" (Wittgenstein, 1953, § 116). This is the essence of ordinary language philosophy, which was inspired by Wittgenstein's PI and the works of John L. Austin, and advanced by Cavell himself. At its core, it appeals to the everyday uses of language as the ground truth. Wittgenstein contrasts this inquiry into the everyday use of language with investigating language as if it were abstracted away from that setting, "when language is, as it were, idling, not when it is doing work" (Wittgenstein, 1953, § 132). Again in NYUA, Cavell remarks that "the behavior of words is not something separate from our lives" (Cavell, 1989, p. 35). Although many of these scholars did not directly interact with each other, there is a continuity between the debates that they engaged in with philosophy and linguistics. One way to frame these struggles at a high level is a nature vs. nurture debate: does language come with a natural

order, or is it shaped by its surroundings? The intent of reducing this debate to nature vs. nurture is to highlight just how broad this problem actually is. We hear of this debate within psychology, biology, and even the nexus of the two in psychopharmacology and medicine: is this person an "addict" naturally/is this substance naturally addictive, or is it a product of their surroundings (is it genetic or is it *epigenetic*?).

§ 2.3.3 Applications in Computational Linguistics

Now that we see how widespread this debate is, let us turn our attention back to linguistics, specifically within the field of computational linguistics. While formalist linguistics views syntax as being internal and autonomous from communication, where syntactic evaluation is constant and the rules of possibilities are predefined, the functionalist takes everyday speech as the starting point, with communication and other external forces being the key factors for language development. To a functionalist, syntax *emerges* through repeated patterns in language use.

Over the past fifteen to twenty years, this functionalist approach to language has seeped its way into computational linguistics and language modeling, even if not explicitly designed to do so. For instance, modern large language models exhibit a proficient level of knowledge of natural language syntax. They do so having only been trained on text alone in a sequential manner. So, much like the functionalist, an LLM learns syntactic rules (or rather, they emerge) solely from the patterns observed in a vast amount of linguistic data. Importantly, for the functionalist, this is done without the need for a separate syntax mechanism. On top of the emergence of syntax from lots of data, words and sub-words in language modeling are represented by dense continuous vectors, rather than discrete atomic symbols. Atomic symbols would be more akin to the formalist approach where they are processed by explicit formal rules of grammar. This runs parallel to the ideas found in analytic philosophy, logical positivism, and early philosophy of language like Wittgenstein's *Tractatus-Logico-Phlosophicus*.

Dense continuous vectors, on the other hand, can capture the statistical patterns that are latent in language use, so syntax is emergent in language use rather than innate to the human cognitive faculties. The use of these vectors can be traced back to distributional semantics, which rests on the distributional hypothesis: "Lexemes with similar linguistic contexts have similar meanings" (Lenci, 2018, p. 152). Given that functionalism appeals beyond formal rules to explain why language is the way it is, an extremely functionalist point of view could be taken up in arguing that words convey the meaning they convey purely because of the other words around them (words that share their distribution). Importantly for the functionalist, this is done without explicit rules. So, the idea of using dense word vectors to encode linguistic information sits closely with some functionalist ideals.

An basic method to construct these word vectors (known as embeddings) is to tabulate a word co-occurrence matrix. The matrix is square, with each row and column representing a word in the vocabulary of a corpus. A "slide" is then applied to each sentence in the corpus where, for each current word, the n words to the left and the n words to the right of it are counted as co-occurring with it. So for the row in the matrix corresponding to the current word, each column for all of the co-occurring context words is incremented. Once this sparse matrix is populated, dimensionality reduction techniques from linear algebra are applied to reduce these vectors to reasonable sizes. This is also done to exchange a sparse matrix full of mostly zeros for a dense vector of mostly real numbers. From these vectors, semantic and syntactic information is encoded all from the words' distributions in language. It is worth noting that modern word embeddings are obtained in a bit more sophisticated manner that encodes context, but nonetheless the idea is the same: linguistic information is baked into a list of numbers. As mentioned previously, the atomic symbols of formalism sit alongside Ludwig Wittgenstein's early work, the *Tractatus-Logico-Philosophicus*. However, the sparse matrix that is created in the slide process for generating static word embeddings is very akin to his later work in PI. There, he writes of words as having "family resemblances"

to each other, where they connect to some words in some ways, and to other words in other ways (Wittgenstein, 1953, § 67). This creates a dense network of words and their associated meanings. So, the formalist-functionalist split has an analog in philosophy, one that nearly perfectly bisects a single scholar's work into two distinct parts. This web-like network of words is also found in psycholinguistics studies based on human attested word associations (De Deyne et al., 2018). So, the view of language as a connected graph where words share properties with each other (syntactic, semantic, morphological, and phonetic) pervades philosophy, psycholinguistics, and computational linguistics.

Much like how the formalist-functionalist counterpart in philosophy can be traced back to the work of one or a few scholars, something similar can be said for that in linguistics. The distributional hypothesis is often attributed to Zellig Harris (Harris, 1954). Harris, ironically, was Noam Chomsky's advisor during his PhD at the University of Pennsylvania. So, while Harris is not explicitly a functionalist, functionalist ideas can be traced back to him, and with that we can see how formalism and functionalism tie together. In fact, all neural language models still utilize dense vector representations of words, even those that are explicitly trained to model language in line with formalist views (i.e. RNNGs).

§ 2.4 Knowledge Distillation

In section 2.1, we went to great lengths to articulate the importance of data efficiency. Data efficiency is certainly not a new concern in deep learning, especially with language modeling. For example, there are many languages that are simply spoken by fewer people around the world, and therefore have less textual data to serve during training. Knowledge distillation (KD) provides a simple and intuitive way to boost training performance in data scarce training environments.

§ 2.4.1 KD Essentials

In deep learning, KD can be thought of as a method of model compression. In fact, Hinton et al. (2015) took direct inspiration from a paper entitled "Model compression" (Bucila et al., 2006) when they published on the topic. In essence, there are two (or potentially more) models present in the training pipeline: one that is actually being trained and updated (the student model), and one that is being used to influence the student model (the teacher model). The goal is to get the student model to mimic the output of the teacher model. Ideally, this allows one to essentially shrink the size of the teacher model—which is usually significantly larger and more cumbersome to train—by compressing its knowledge into the smaller student model.

On top of this reduction in model size, the student model also benefits from more signal than what the data alone provides. A well-trained teacher can provide more information in its output probability distribution than the pure ground truth labels can. Due to this added signal, a model can learn more from less data. Take an example of KD from computer vision: the rise of success with transformer models in NLP prompted researchers to implement them for computer vision tasks too. Dosovitskiy et al. (2021) was able to achieve competitive results on the ILSVRC-ImageNet1k benchmark with their ViT model (Vision Transformer). However, the authors note that when pre-trained on just the training set for ImageNet (1.3M images), ViT underperforms. In order to bring ViT performance up to par with some of the best convolutional neural nets, it had to be trained on the JFT-300M dataset, which contains 300M images, and is proprietary to Google. To address this data-inefficiency, Touvron et al. (2021) adopted nearly the same model architecture as ViT, but implemented a KD procedure. The result was an astonishing reduction in training data required: their model (which they called DeiT - Data efficient image Transformer) achieved an accuracy on ImageNet1k of 85.2% while only being trained on ImageNet's 1.3M training images. For comparison, a similarly sized ViT model required 300M training images to reach 84.15%

accuracy. It is worth mentioning that the DeiT authors also applied various augmentations to their training images, which effectively created more data with slight adjustments. One aim of the present research is to see if the KD approach alone yields significant benefits for language modeling, seeing that tokens cannot be augmented nearly as much as images can be.

In terms of how the student model is encouraged to mimic the teacher model, this can be implemented in a couple of different ways. In a non-KD training environment, one is already trying to encourage the model to mimic the ground truth targets. This is done by comparing the output of the model to the target values via some kind of objective/loss function. Gradient-descent is then performed, searching for a solution (set of weights in the model) that minimizes this loss function (i.e. makes the output of the model closely resemble the ground truth targets). KD is not much different, except instead of comparing the predicted outputs of a model to the ground truth, these outputs are compared to the outputs of the teacher model. Oftentimes, both approaches are combined into one loss function, and a weight is assigned to each term. This weight is often set by a hyperparameter, alpha, such that one part of the loss is multiplied by alpha, and the other part is multiplied by (1 - alpha).

The distillation term in the loss function (the one comparing the outputs of both models) can be implemented in a number of ways. Hinton et al. (2015) experiment with using 2 cross-entropy terms to their loss function, one comparing the output logits of the student model to the ground truth labels, and another comparing the output logits of the student model to the predicted labels from the teacher model. They call this hard distillation. They also experimented with replacing the cross-entropy term between the two models with the Kullback-Leibler (KL) divergence between the two models' probability distributions. KL divergence essentially measures the disparity between two probability distributions, and the approach is commonly used for KD (Kuncoro et al., 2019). Hinton et al. (2015) call this

form soft distillation, where the student model has access to the teacher's entire output probability distribution, and not just the predicted labels. Ideally, soft distillation provides more information to the student model by showing how closely the teacher may perceive two classes to be to each other; however, hard distillation with just the predicted labels from a teacher has also shown to be effective in certain situations (Touvron et al., 2021).

In certain cases where the teacher model is able to predict the correct label with extremely high confidence, providing the student model with this distribution does not add much more information. This happens with the MNIST dataset of handwritten digits often, as Hinton et al. (2015) point out. Therefore, in order to pull out the underlying similarities between classes, the model's logits are often scaled by a temperature parameter. The effect of this is a flattening out of the distributions, so that classes that are predicted with the second, third, etc. highest confidence are shown to be closer to the actual predicted class, allowing for more information to pass from the teacher model to the student model.

§ 2.4.2 Syntax to Sequence KD

Up to this point we have covered a swath of different topics, and hopefully one can see how they are tied together in this research project. Current state of the art language models are incredibly data inefficient. These inefficiencies pose their own questions and concerns, as described in section 2.1. Different types of language models, while having their own issues, provide unique internal representations of language. Specifically, RNNGs can encode syntactic structure more explicitly, whereas sequential models alone cannot.

Given that knowledge distillation is a well-known deep learning technique used for reducing model size and the amount of training data needed for convergence, we have all of the necessary conditions for setting up a distillation pipeline. In the next section, we detail a KD setting that uses an RNNG as a teacher model and an autoregressive transformer as the student model. Using a significantly smaller training dataset (less than 100M words) than

those typically used to train LLMs, our hopes are that we can address both the practical concerns associated with the costs of training LLMs, as well as the scientific concerns surrounding why they require so much data to acquire language, while humans do not. While Kuncoro et al. (2019) performed similar experiments using an RNNG model as a teacher in KD, their student model was an LSTM language model, not a transformer. LSTMs are some of the best known RNN implementations for language modeling, but they nonetheless fail to capture long-range dependencies quite like transformers do. They also *only* trained the models on the Penn Treebank (Marcus et al., 1993), which is small (roughly 1M words) and only covers a subset of English linguistic phenomena. Kuncoro et al. (2020) also used an RNNG as a teacher to distill syntax into a transformer; however, the transformer model was an encoder only BERT based model, meaning it received context bidirectionally. In our case, we will be using an autoregressive transformer like the GPT series, so it only receives context from previous tokens, rather than bidirectionally. Even more important to the concerns of this project, we restrict our largest dataset to be less than 100M words. In comparison, the pre-training data used to train the BERT student model in Kuncoro et al. (2020) was the same as the pre-training data used for the original BERT model, sitting at 3.3B words (Devlin et al., 2019). As we will see in section 3.1.3, the BabyLM corpus provides a nice balance between being a cognitively-plausible size, and offers diversity in the types of language one may encounter.

Chapter 3

Data and Procedures

In this section we will describe the data and procedures used to set up a knowledge distillation training pipeline with an RNNG as the teacher model and a transformer as the student model. Alongside the distilled transformer, we also train a non-distilled transformer to serve as a control. The goal of this experimental setup is to see whether the distilled transformer has a better understanding of natural language than the non-distilled transformer. To assess the models' understandings of natural language, we evaluate both of them on the Benchmark of Linguistic Minimal Pairs (BLiMP) (Warstadt et al., 2020).

§ 3.1 Training Datasets

Multiple datasets were used to train the models in this experiment, namely the Penn Treebank (Marcus et al., 1993), the BLLIP corpus (McClosky et al., 2008), and the BabyLM corpus (Warstadt et al., 2023).

§ 3.1.1 Penn Treebank

The Penn Treebank (PTB) is a staple in any NLP practitioner's back pocket. This dataset is the same one used to train the RNNGs in Dyer et al. (2016), as well as in Kim et al. (2019). The dataset is hand annotated by human linguists, and it stems from roughly 2,500 Wall Street Journal (WSJ) articles over a three year span in the late 1980s.

Compared to modern NLP corpora, the PTB is tiny, containing only a little more than

1 million words in its syntactically annotated portion. Consequently, the initial results with this dataset were not very telling: the corpus is simply too small for modern LLM architectures to really grasp the fullness of linguistic knowledge, even with the help of a syntactic model as its teacher. The only consistent patterns in the dataset are that periods tend to come at the end of a sentence followed by the end-of-sequence (EOS) token `</s>`. So, even after good perplexity metrics during training, all this model learned to do when prompted to generate new text was predict a period and EOS token, even if the input text was a sentence fragment such as "The United States is . . . " . Given this, the PTB was primarily only used for preliminary model testing. This did come in handy when architectural changes were made to the model, or when new optimizers or learning rate scheduler configurations needed to be tested.

§ 3.1.2 BLLIP Corpus

The second dataset used was the BLLIP (Brown Laboratory for Linguistic Information Processing) corpus (McClosky et al., 2008), which is similar to the PTB in that it pulls articles from the WSJ in the late 1980s. However, the sentences in this corpus were annotated using automatic part of speech taggers and parsers. So, while the PTB offers the reliability of human annotated sentences, it lacks volume to train modern LLMs. The BLLIP corpus, on the other hand, exchanges that reliability of human annotation with silver grade, automatically annotated trees, but an increase up to almost 40 million words.

§ 3.1.3 BabyLM Corpus

The BabyLM corpus is a collection of other sub-corpora that depict multiple different linguistic environments. It contains roughly 95M words in its training set, which is more than twice the amount found in the BLLIP corpus. Additionally, the diversity of corpora introduces more types of speech and writing, as opposed to the formally written WSJ articles found in BLLIP. Our initial thoughts were that this diversity would provide more scientific

interest as it accounts for the types of language a human sees across their entire lifespan. However, as we will come to see later on, this diversity also poses its challenges.

The corpora that make up the 2024 BabyLM challenge dataset are listed below. In many cases, only subsets of these corpora were used to keep the number of words restricted to less than 100M.

- CHILDES (Macwhinney, 2000)
- Switchboard (Stolcke et al., 2000)
- Simple Wikipedia (Wikimedia Foundation, 2003)
- Standardized Project Gutenberg Corpus (Gerlach and Font-Clos, 2018)
- Open Subtitles (Lison and Tiedemann, 2016)
- Spoken British National Corpus (BNC) (BNC Consortium, 2007)

§ 3.2 BLiMP Evaluation

The BLiMP evaluation suite is one of the most comprehensive tests to date to evaluate the linguistic knowledge of a language model. It consists of 67 tests that each highlight a unique contrast for a given linguistic phenomenon (note that the original BLiMP publication uses the term **paradigm** to talk about a specific contrast within a linguistic phenomenon; here we will use the generic term **test** to avoid confusion with other technical terms in linguistics). Each test contains 1,000 minimal pairs to highlight that contrast, and they can all be grouped into one of twelve broader categories of linguistic phenomena. All phenomena are regularly covered in syntax courses and/or textbooks on generative syntax (Sag et al., 2003; Adger, 2003; Sportiche et al., 2013).

In this evaluation, a minimal pair is a pair of two sentences that differ by only one word (with exceptions made for compound phrases like *fewer than* and *at most* for quantifiers in

Table 3.1). So, in the evaluation pipeline, a model is considered to be correct when it assigns a higher probability to the grammatical sentence than it does to the ungrammatical sentence for each minimal pair. In essence, it is an exam with 67,000 true-false questions. [1]

Table 3.1: Table 2 from Warstadt et al. (2020) describing the 12 different linguistic phenomena included in BLiMP. N is the number of tests (unique contrast) that each phenomena contains in the dataset.

Phenomenon	N	Acceptable Ex.	Unacceptable Ex.
Anaphor Agreement	2	*Many girls insulted themselves.*	*Many girls insulted herself.*
Argument Structure	9	*Rose wasn't disturbing Mark.*	*Rose wasn't boasting Mark.*
Binding	7	*Carlos said that Lori helped him.*	*Carlos said that Lori helped himself.*
Control/Raising	5	*There was bound to be a fish escaping.*	*There was unable to be a fish escaping.*
Det.-Noun Agreement	8	*Rachelle had bought that chair.*	*Rachelle had bought that chairs.*
Ellipsis	2	*Anne's doctor cleans one important book and Stacey cleans a few*	*Anne's doctor cleans one book and Stacey cleans a few important*
Filler-Gap	7	*Brett knew what many waiters find.*	*Brett knew that many waters find.*
Irregular Forms	2	*Aaron broke the unicycle.*	*Aaron broken the unicycle.*
Island Effects	8	*Who has Colleen aggravated before kissing Judy?*	*Who has Colleen aggravated Judy before kissing?*
NPI Licensing	7	*The truck has clearly tipped over.*	*The truck has ever tipped over.*
Quantifiers	4	*No boy knew fewer than six guys.*	*No boy knew at most six guys.*
Subject-Verb Agreement	6	*These casseroles disgust Kayla.*	*These casseroles disgusts Kayla.*

There are several motivations for evaluating the resulting models on the BLiMP benchmark. For one, since the training corpora and motivations of this project are largely taken from the BabyLM challenge (Warstadt et al., 2023), it is only fitting to also evaluate the

[1] Table 3.1 is a direct copy from Warstadt et al. (2020) with the exception of the examples for Island Effects. The original BLiMP publication used left branch extraction examples here, which are grouped under island effects, though not actually attestations of this phenomenon.

models in the same way with BLiMP. It is also fairly comprehensive, covering a wide range of topics that require a deep understanding of the structure of language. Lastly, the benchmark is very simple and easy to use: all one has to do is pass a minimal pair to the model and see which sentence it assigns a higher probability to. If it assigns a higher probability to the grammatical sentence, then the model is correct. If it assigns a higher probability to the ungrammatical sentence, then it is incorrect. In the end, the overall accuracy of the model can be judged by the number of correct predictions it made compared to the number of total predictions made. Although accuracy is certainly not the end-all-be-all of validation metrics for language modeling, we believe the BLiMP tests are granular enough to provide insights into the specific linguistic patterns that a model has learned. These tests provide a much more thorough view of what a language model has learned compared to perplexity. Perplexity is simply the exponentiation of the negative log likelihood loss (NLL loss). NLL loss is a common criterion in multi-class training environments in deep learning. At every prediction point, the model has to predict which token comes next in a sequence. Since tokens are discrete values, language modeling is a prime example of multi-class training, in contrast to predicting a continuous numerical value. So, while NLL loss and perplexity are common metrics to watch during training to see if the model is learning *something* (i.e. converging), they tell researchers nothing about *what* the model is learning. Hopefully, by assessing the model's accuracy on various linguistic phenomena, we can further see what aspects of language it "knows" well and which ones it does not yet "understands".

§ 3.3 Procedures

§ 3.3.1 Data Preprocessing

In order to train a language model in a KD setup with an RNNG as the teacher model and a transformer as the student model, the datasets had to be processed specifically to the needs of both models. In this way, the data can be given to both models during the training

pipeline in order to compare the output distributions together. For the teacher model, we used the RNNG implementation and data preprocessing scripts from Kim et al. (2019). Since the original RNNG implementation operates at the whole word level, but the transformers operate at the token level, these preprocessing scripts had to be modified to handle tokenized sentences. Before passing the parsed sentences to the scripts for preprocessing, each word in the parse tree had to be tokenized. So, rather than the terminals in the parse tree being whole words, they were split into sub-word tokens. Each sub-word token was left in its original node in the tree though, so the only change to the list of parsing actions is that a `SHIFT` is added to the parse stack for every token found, rather than every whole word found. For all corpora used, we fit a Byte-Pair Encoding (BPE) tokenizer on the raw text from each corpus. A modest vocabulary size of 12,000 tokens was used for every tokenizer.

At a high level, this script takes in syntactically parsed sentences in the same format as the PTB and BLLIP corpus and does the following:

1. Sifts through each sentence to obtain its shift-reduce parsing actions. This is done by noting at what point constituents are created (i.e. when an open parenthesis is found), when sub-word tokens are found, and when constituents are closed (when a closing parenthesis is found). Open parentheses are noted with `NT(` for "opening a non-terminal", sub-word tokens are given a `SHIFT` action, and closing parentheses are given a `REDUCE` action. `NT(` and `SHIFT` are when elements are added to the stack individually, while `REDUCE` is when these elements are combined into one complete constituent to be added back onto the stack. This sequence of actions serves as the ground-truth parsing actions for each sentence.

2. Sequences of just the sub-word tokens are also gathered, as though the data were flat sentences with no structural annotations. These sequences of tokens are then converted to their respective token IDs, which are used as input to the language models to predict

the next tokens.

Since the PTB and BLLIP corpus are already parsed, all we had to do was tokenize their parse trees before passing them to the preprocessing scripts. The BabyLM corpus, however, was not parsed or tokenized in advance. In order to obtain syntactically parsed versions of the BabyLM corpus, we used the spaCy implementation of the Berkeley neural parser (Kitaev and Klein, 2018). Due to the slow parsing process, most of the sub-corpora in the dataset had to be chunked into subsets so they could be parsed simultaneously. This took about a week to complete.

§ 3.3.2 Model Architecture

As previously noted, the RNNG implementation used is the same as the one used in Kim et al. (2019). As for the student transformer model, it roughly mimics the same architecture as GPT-2, so it is a Decoder-only model. In the wake of recent research on training language models with scarce amounts of data, some modifications were made. Specifically, two adjustments were adopted following the LTG-BERT architecture from Samuel et al. (2023): the use of a GeGLU activation function in the feed-forward modules, as proposed in Shazeer (2020), and the use of additional layer normalizations inside the decoder blocks of the transformer following Shleifer et al. (2021).

The GeGLU activation function is an adaptation to the original Gaussian Linear Unit (GLU) activation function, which was proposed in Hendrycks and Gimpel (2023). The PyTorch implementation for GeGLU is show below:

```
def geglu(x: torch.Tensor) -> torch.Tensor:
    x, gate = x.chunk(2, dim=-1)
    return x * F.gelu(gate)
```

The original transformer from Vaswani et al. (2017) uses layer normalizations (LN) *after* (post-LN) the residual connections for each sublayer (attention and feedforward modules),

while many modern LLM architectures use LN *before* each sublayer (pre-LN), including GPT-2 (Radford et al., 2019). Shleifer et al. (2021) propose what they call the Normformer, where they use both pre-LN and post-LN. This was motivated after observing that, while pre-LN improved the stability and convergence of models during training, the "gradients at earlier layers tend to be larger than gradients at later layers" (Shleifer et al., 2021). The use of both pre-LN and post-LN helps to balance out this gradient mismatch. Figure 3.1 displays the final architecture for the decoder blocks used in the models for this experiment.

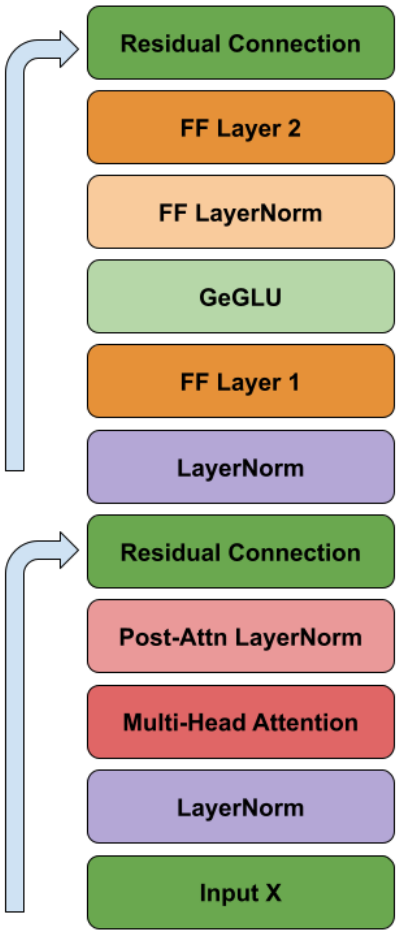

Figure 3.1: Decoder Block flowchart

A number of configurations for the transformer models were tested. The final hyperparameter configurations are shown in Table 3.2.

Table 3.2: Hyperparameters used for transformers

Hyperparameter	**Value**
Seed	3435
Embedding Dimensions	1536
Feedforward Dimensions	3072
Embedding Dropout	0.1
Feedforward Dropout	0.1
Attention Dropout	0.1
# Decoder Blocks	48/72
# Attn Heads	12
Initial Learning Rate	5e-5
Alpha	0.6

§ 3.3.3 Training

Model training configurations varied to a certain extent depending on the available resources. All models were trained using the University of Georgia's Sapelo2 research computing cluster. Longterm GPU computing nodes (8-30 days max) weren't always available, but in general, it wasn't always necessary to use longterm compute nodes. Most models tended to converge on a given perplexity within 7 days. 4 Nvidia A100 GPUs were used for every training run of the full BLLIP and BabyLM corpora. This was implemented using PyTorch's Distributed Data Parallel (DDP) module. Batch sizes of 16 were used. We attempted to use larger batch sizes, but the 80GB of VRAM on the A100 GPUs couldn't seem to hold the models, larger batches, and all of the intermediate activation values. Due to the RNNG requiring parsing actions, each batch of data contained more than just the input sentences taking up space in the GPU RAM. The transformers came out to 1.15B and 1.72B trainable parameters for the 48 and 72 block models, respectively.

Prior to training, the entire validation dataset was passed to the models to gather initial baseline metrics. Once training began, the validation set was run after every epoch. If a new best (lowest) perplexity was attained, then that copy of the model was saved.

To optimize this model, we used the AdamW optimizer (Loshchilov and Hutter, 2019), along with a sequential learning rate (LR) scheduler. This sequential LR scheduler consisted of a linear warmup, followed by a cosine annealing with warm restarts scheduler. The learning rate increased linearly from 0 to 5e-5 during the first 5% of training batches. Once reaching 5e-5, the learning rate decayed after every batch for the first 8 epochs according to the cosine annealing scheduler, with an absolute minimum learning rate of 1e-9. After the 8th epoch, the learning rate restarted at 5e-5, and its decay cycle was doubled to 16 epochs (though we never trained the largest models beyond 12 epochs). The loss function during distillation is shown below:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{NLL}} + (1 - \alpha) \cdot \mathcal{L}_{\text{KL}}$$

Chapter 4

Results

All models were evaluated on perplexity during training and BLiMP after training. Perplexity (PPL) is simply the exponentiation of the average negative log-likelihood loss (NLL loss) per token, which is the loss function that all models are trying to minimize during training. Perplexity provides a more interpretable validation metric as it can be seen as a proxy for the number of choices a model is "perplexed" by (or considering) when making a prediction. In both NLL loss and PPL, lower is better.

§ 4.1 Training Results

Factors that play into a model's perplexity on the validation data are primarily the model size, the training dataset size, and the dataset diversity. In every case, the syntactically distilled models achieved lower perplexities than their corresponding control models. They also achieved lower perplexities than their teachers in every case except for the PTB. This indicates that, to some degree or another, the syntactic knowledge from the RNNG is being transferred to the student models and helping them converge more quickly. All model perplexities are shown in Table 4.1.

The disparity between control and distilled PPLs is most clearly observed with the PTB trained models. Given that the PTB is such a small dataset, it makes sense that neither the student nor the control transformers achieved PPLs as low as the teacher RNNG. This model explicitly models syntax and next-token predictions, whereas the transformers, even

Table 4.1: Validation perplexities (PPL) for all models trained on all 3 corpora. Epochs refer to when the control and distilled models reached their lowest values, respectively. The * denotes that the RNNG used during distillation for the sub-corpus BLLIP models was trained on the full BLLIP corpus of about 40M words.

Dataset	**No. Decoder Blocks**	**Control**	**Distilled**	**Epoch Reached**	**RNNG**
PTB	12	96.17	90.21	37, 40	52.86
BLLIP 2.5M	12	66.56	61.44	34, 39	44.29*
BLLIP 5M	12	42.33	40.48	29, 33	44.29*
BLLIP 10M	12	30.45	29.61	31, 29	44.29*
BLLIP	48	16.30	16.03	9, 10	44.29*
BabyLM	48	30.91	30.30	8	63.71
BabyLM	72	31.14	30.69	8	63.71

the distilled ones, are only performing next-token predictions. So, the RNNG learns a deeper representation than the transformers. There is simply not enough data in the PTB for a model to sufficiently learn a probability distribution over a vocabulary from a purely sequential representation of language.

The BLLIP trained models achieved the lowest PPLs when trained on the full dataset. These models also outperformed the RNNG by a wide margin. This is likely due to the nearly forty-fold increase in number of words when moving from the PTB to the BLLIP corpus.

The BabyLM trained models did not reach perplexities as low as the BLLIP corpus, but nonetheless similar patterns were found. Both transformer models achieved lower PPLs than the RNNG trained on this corpus, and the distilled model achieved a lower perplexity than its control counterpart.

The difference in validation PPLs between the BLLIP and BabyLM models is slightly surprising since the BabyLM corpus has over 90M words in it compared to BLLIP's 40M. However, it is important to remember that the BabyLM corpus is really a collection of 6 different subsets of corpora, while the BLLIP corpus is simply Wall Street Journal articles

from the late 1980s. This means the content and vocabulary in the BLLIP corpus is relatively homogeneous compared to that of the BabyLM corpus. The BabyLM corpus also includes transcriptions of spoken language, such as the subset from CHILDES (Macwhinney, 2000). This form of language is much less structured than the kind found in written newspaper articles such as the WSJ. So, even though these sentences were parsed in a similar manner to the BLLIP corpus, the variety of different parses is much higher in the BabyLM corpus since the transcriptions might include filler words such as "umm", sentence fragments, or repeated phrases.

Due to the sharp decline in PPLs when moving from the PTB to BLLIP (and due to the closing gap between the control and distilled PPLs in doing so), we also trained control and distilled transformers on subsets of BLLIP. Keeping the model size the same as the PTB, our aim was to see at what number of training words did the models' PPLs begin to decrease in line with the larger models trained on more data. Beginning at 10M words, these models reached PPLs in the low 30s. With only a 10x increase in the number of training words, these models already began to converge on PPLs close to the full BLLIP trained models, as well as the BabyLM models. From there, we cut the dataset in half to 5M words, and eventually 2.5M words. As expected, PPL increased as the number of training words decreased. This steep drop in PPL over the first 10M training words coincides with the findings of Zhang et al. (2021). There, the authors pretrained language models on 1M, 10M, 100M, and 1B words. While PPL was not reported for these models, their scores on the BLiMP evaluation were. 1M words saw poor BLiMP accuracies, but moving minimally beyond that results in a steep increase in performance (see Fig. 1 in Zhang et al. (2021)). With these findings, and with the sharp drop in PPLs that we report, one might conclude that explicitly modeling syntax (or distillation from explicit modeling) is unnecessary to capture syntactic structures in language. This may in fact be the case; however, section 4.2 will show us that a deeper knowledge of syntactic structures through distillation nonetheless

improves a model's understanding of language as a whole.

§ 4.2 BLiMP Evaluation

As mentioned in section 3.2, perplexity really only tells us that a model is converging at all, i.e. it is "learning" *something*. The BLiMP evaluation provides a quantitative way to measure a model's competence with respect to a number of specific linguistic phenomena.

With the exception of the PTB and BLLIP subset trained models, each set of transformers (control and distilled) received the entire BLiMP evaluation. For each prediction, the probability that the model associates with the correct next token is gathered and converted to a log-probability, which can then be summed up for the entire sentence. The model is considered correct when it generates a higher log-probability for the correct sentence. The following tables show the final accuracy scores grouped by the 12 linguistic phenomena for each model.

§ 4.2.1 BLLIP models' performance on BLiMP

Table 4.2: Accuracies on the BLiMP evaluation test for the transformers trained on the BLLIP corpus.

Linguistic Phenomenon	**Control Accuracy**	**Distilled Accuracy**
Anaphor Agreement	67.35	72.45
Argument Structure	64.03	63.04
Binding	67.67	70.89
Control Raising	62.00	62.46
Determiner Noun Agreement	76.08	77.98
Ellipsis	49.30	53.75
Filler Gap Dependency	60.83	59.66
Irregular Forms	78.25	74.15
Island Effects	48.05	50.36
NPI Licensing	68.34	68.61
Quantifiers	64.73	64.73
Subject Verb Agreement	69.17	69.60

The BLLIP trained models are shown in Table 4.2. Here, the syntactically distilled

model outperformed the non-distilled control model on 8 of the 12 phenomena. The control outperformed the distilled model on 3 phenomena. They perfectly tied on 1: Quantifiers.

To look at these models at a more granular level, we can observe their performances at the individual test level (so, not grouping by linguistic phenomenon; see table in Appendix A). At the individual test level, the control model scored a higher accuracy than the distilled model on 28 tests, while the distilled model scored a higher accuracy than the control model on 38 tests. They perfectly tied on exactly 1 test: Matrix Question NPI Licensor Present, both with stellar scores of 93.9%.

I find it worth wondering how much one model outperforms the other on specific tests. Are there any that one model learned exceptionally better than the other model? The max difference where the distilled model outscored the control model was 8.5%. This test has to deal with binding, specifically Principle A reconstruction.[1] Conversely, the max difference where the control model outscored the distilled model was 6.0%. This test has to deal with island effects.

When the control model outscored the distilled model, it did so by an average of 2.33%. In comparison, when the distilled model outscored the control model, it did so by an average of 3.2%.

§ 4.2.2 BabyLM models' performance on BLiMP

Table 4.3 shows the results for the 48 block transformer models trained on the BabyLM corpus. These models share the same number of blocks and total parameters (1.15B) with the BLLIP trained models, but we see a degradation in results between the distilled and control models. Here, the *control* model outperformed the distilled model on 7 out of 12 phenomena, rather than the other way around. At the individual test level (Appendix A), the control model outscored the distilled model on 38 tests, while the distilled model outscored

[1] In binding theory, principle A requires that an anaphor (e.g. reflexive pronouns like *herself*, *himself*, etc.) must be bound, i.e. it must be c-commanded and co-indexed by an antecedent in its clause.

the control model on 28 tests. Once again, they perfectly tied on exactly 1 test, this time being Principle A, Case 1.

Table 4.3: Accuracies on the BLiMP evaluation test for the 48 block transformers trained on the BabyLM corpus.

Linguistic Phenomenon	**Control Accuracy**	**Distilled Accuracy**
Anaphor Agreement	96.05	97.35
Argument Structure	68.14	68.70
Binding	75.17	73.40
Control Raising	71.86	68.92
Determiner Noun Agreement	85.90	87.61
Ellipsis	69.70	66.35
Filler Gap Dependency	69.51	70.71
Irregular Forms	88.20	82.95
Island Effects	66.41	60.95
NPI Licensing	71.66	77.34
Quantifiers	79.75	77.65
Subject Verb Agreement	77.93	76.73

The largest difference at the individual test level when the control model outperformed the distilled model came with island effects, particularly with *Wh*-Island. Here, the control model outscored the distilled by 12.3%. The largest difference when the distilled model outscored the control model was 19.2% (Only NPI Licensor Present).

Even though the distilled model did not outperform the control model very often, when it did, it did so by an average of 4.11%. Conversely, when the control model outperformed the distilled model here, it did so by an average of 3.92%

Given the larger amount of training data in the BabyLM corpus compared to the BLLIP corpus, we retrained the BabyLM models with 72 decoder blocks, which seemed to be about the maximum model size that could fit inside each A100 GPU's memory, along with the batches of data. These models contain about 1.72B trainable parameters. Table 4.4 shows these results, with the distilled model making a comeback and outperforming its control model on 9 out of 12 phenomena.

Table 4.4: Accuracies on the BLiMP evaluation test for the 72 block transformers trained on the BabyLM corpus.

Linguistic Phenomenon	**Control Accuracy**	**Distilled Accuracy**
Anaphor Agreement	95.80	98.35
Argument Structure	69.33	71.97
Binding	73.56	73.61
Control Raising	72.52	71.28
Determiner Noun Agreement	84.49	87.66
Ellipsis	60.05	66.65
Filler Gap Dependency	71.04	72.50
Irregular Forms	81.95	91.90
Island Effects	65.60	65.15
NPI Licensing	74.69	79.20
Quantifiers	74.23	69.98
Subject Verb Agreement	78.05	80.13

With the increase in dataset size and model size, we once again see the distilled model outperform the control model at the individual test level, this time on 42 out of 67 tests. The control model outperformed the distilled model on 23 tests, and they perfectly tied on 2, both achieving accuracies of 99.9%.

The largest difference when the distilled model outperformed the control model came with Irregular Past Participle Adjectives at 17%. The largest difference when the control model outperformed the distilled model came with Superlative Quantifiers 1 at a whopping 26.5% difference.

On average, when the distilled model outperformed the control model, it did so by an average of 4.65%. When the control model outperformed the distilled model, it did so by an average of 3.49%. So, even with the large 26.5% difference on superlative quantifiers, the other margins were not enough to make up the difference when averaging across all margins of victory.

§ 4.3 Text Generation Capabilities

Lastly, we prompted the BLLIP and BabyLM (72 block) models to get a sense for their text generation capabilities. Though not always the most coherent or insightful sentences, in many cases the models were able to generate full constituents, if not complete (grammatical) sentences. Some examples from these models are shown in Table 4.5. Note that while the BLLIP models required no temperature scaling to the logits at each prediction point, the BabyLM models required a temperature scaling of 100.0.

Table 4.5: Text generated by the BLLIP and 72 block BabyLM trained transformers

Model	**Input**	**Completed Text**
Control BLLIP	“ The United States ”	“ The United States , which <unk> is the world’s largest nuclear power , has a policy of* keeping nuclear-powered nuclear-weapons and nuclear weapons from<unk> being used<unk> to protect the U.S. from<unk> the nuclear-free zone .<unk></s>”
Distilled BLLIP	“ The United States ”	“ The United States , in the wake of the Iran-Contra scandal , is a vital link in the Reagan administration’s policies .</s>”
Control BabyLM 72	“ The President of the United ”	“ The President of the United Kingdom is elected every two terms .</s>”
Distilled BabyLM 72	“ The President of the United”	“ The President of the United National Secure Association and Secretary General is a British company who works to ensuring the safe keeping in place of a threat .</s>”

Chapter 5

Discussion

In the previous chapter, we encountered a number of interesting results with respect to training/model convergence, as well as in targeted domain evaluations.

Increasing the dataset size certainly proved helpful for model convergence, as we saw a drastic drop in validation PPL when transitioning from the PTB to the BLLIP subsets, and ultimately the full corpus. Going from BLLIP to BabyLM, however, we actually saw an *increase* in perplexity, despite the BabyLM corpus containing more than twice the amount of words in BLLIP. This increase in PPL can most probably be explained by the increase in diversity that comes with the BabyLM corpus. As previously mentioned in section 4.1, the BLLIP corpus contains 40M words all stemming from one source: the Wall Street Journal. Here, we have a relatively homogeneous content, where articles are almost always concerning business, finance, economics, or politics. In contrast, the BabyLM corpus contains over 90M words stemming from subsets of 6 other corpora, all of which are fairly diverse. The CHILDES subset contains written transcriptions of child directed speech; Switchboard contains written transcriptions of telephone conversations; SimpleWiki and Gutenberg probably contain the most edited and syntactically structured text; and Spoken BNC and Open Subtitles also contain written transcriptions of speech, although much of this is more longform and syntactically structured compared to CHILDES and Switchboard. This increase in diversity among the different subsets used in the BabyLM data poses a challenge for models during training to be able to account for the wide distribution of words that it may encounter.

With this challenge in place, we still saw consistent patterns where the syntactically distilled models reached lower PPLs than their control counterparts. Even when the student models are significantly larger than their teachers (which is uncharacteristic in knowledge distillation), knowledge can still be transferred from the teacher to the student model. This is evident by the fact that the distilled model tended to outperform the control/non-distilled model on many BLiMP tests. The caveat to this is that, at a certain amount of training words, an increase in model size has to accompany the increase in dataset size. When we increased the dataset size, but kept the model size the same, we saw less of an effect from distillation. While the aims of this project are to explore small model training with small dataset sizes, we still limited the number of words in the training dataset to a cognitively plausible amount of less than 100M, and the total number of trainable parameters in our largest model is only 1.72B. This is significantly smaller than most industry grade LLMs.

Model to model comparisons aside, we see strong improvements on BLiMP for both the distilled and control models when moving from the BLLIP corpus to the BabyLM corpus. Take anaphor agreement: with that phenomenon alone we saw a 25-30% improvement for both models with the increase in data.

In general, the average test accuracy score for the BLLIP trained control model was 64.5%, and for the distilled model it was 65.3%. The increase in data from the BabyLM corpus brought these average accuracies up to 74.6% for the control model and 74.1% for the distilled model - a 9-10% overall accuracy increase. Increasing the model size brought the overall accuracy up some for the distilled model, but not by much (75.8%, a roughly 1.7% improvement). For the larger control BabyLM model, we actually saw overall accuracy drop to 74.1%, a decrease of 0.5%.

Over-fitting might be the best possible explanation to account for the reduced performance from the 72 block control BabyLM model. To evaluate potential over-fitting in more detail for both the controlled and distilled 72 block models, we tracked performance decreases

for all 67 tests. For the control model, when moving from 48 to 72 decoder blocks, we saw accuracy scores decrease in 42 of the 67 tests. In comparison, we saw a decrease in accuracy scores for the distilled models in only 19 of the 67 tests when increasing the number of decoder blocks. Of those 19 tests, 14 of them were also found for the control models, meaning that both models saw decreases in accuracy for them as model size went up.

Excluding the 14 tests where both models saw decreases in accuracy, this leaves 28 that decreased only for the control model when scaling up. Conversely, there were only 5 tests that saw accuracy decreases for just the distilled models. Of the 28 that decreased only for the control models, they decreased by an average of 3.28%. For the 5 that decreased for only the distilled models, they decreased by an average of 0.90%.

With this all in mind, how can we account for the large jump in BLiMP performance for the distilled model when scaling up the model size? Is it the case that the increased number of decoder blocks brought about this performance on its own? This seems unlikely, because otherwise we would have also seen an improvement for the control model when scaling its size up. The fact that we did not see this improvement indicates that the marriage between the increased model size and the signal from the syntactic teacher model helped the distilled model. Its overall accuracy increased, as did the number of tests that it outperformed the control model on. Simply scaling up the number of trainable parameters alone does not help - these models need something to encode in these trainable parameters as well. The distilled model used these extra parameters to encode better syntactic representations. While this might not be the case for every single test, given that the distilled model got worse on 19 of them, the control model also got worse on 14 of these 19. So on only 5 tests did the distilled model see a degradation by itself, compared to the 28 tests that the control model got worse on by itself. Once more, the control model saw a decrease in its overall accuracy as its model size went up, while the distilled model saw an increase in its overall accuracy. So, it might be safe to say that the distilled model made good use of the syntactic signals it

received from the teacher, even when the teacher is a much smaller model itself.

§ 5.1 Does Knowing the Tree Help?

There were a few instances where models scored significantly below random chance. Therefore, this discussion would not be complete if we did not address where these models performed horribly within the BLiMP evaluation. Looking at the results in Appendix A, the first set of alarmingly low scores occurs with Binding, specifically principle A reconstruction. As previously mentioned, principle A states that an anaphor (reflexive pronoun like *herself*) must be bound in its clause. The reconstruction test includes examples like *It's herself who Karen criticized* vs. *It's herself who criticized Karen*, with the former being the grammatical example. The trick with the grammatical example is that the anaphor comes before its antecedent in the surface structure (but not the deep structure). So, one must reconstruct this sentence back to its original form to see the difference: *Karen criticized herself.* As for *It's herself who criticized Karen*, *Karen* is the object of *criticized* in this sentence, not any kind of subject. So, *herself* must be the subject in this sentence, except this means we have an unbound anaphor. This violation of principle A is the reason why this example is ungrammatical. Interestingly enough, the control and distilled BLLIP-trained models scored 64.1% and 72.6%, respectively, for this test. In the original BLiMP paper results, no model scored above 46% on this test, and the average human score was only 78% (making it the ninth worst test out of 67 with respect to human scores). This high performance from both BLLIP-trained models - above both the BLiMP baselines and the BabyLM model scores - indicates that these syntactic structures are much more well attested in the BLLIP corpus; the 8.5% increase from the control to distilled scores shows that knowledge of the syntactic structures from the RNNG for these examples does indeed help.

There are two other tests that we saw models severely struggle with. Both of these came within the Filler Gap Dependency phenomenon, specifically *Wh-* vs *That* with Gap and *Wh-*

vs *That* with Gap Long-Distance. The two tests are the same, but with more intervening words between a subject and its verb in the long-distance examples. For both of these tests across all models, none of them scored above 28.9%. In the original BLiMP results, the highest any model scored on these tests was GPT-2 at 56%, with humans scoring only as high as 77%. Even after "building as large and diverse a dataset as possible" (Radford et al., 2019), GPT-2 scores barely above chance on these tests. The fact that the average human score on these tests was 77% and 75% for short/long distance, respectively, tells us that these distinctions are inherently hard to grasp. This makes intuitive sense; *Nina has learned that most men sound like* resembles the beginning of a grammatical sentence... if there were more words to come after *like*. Complementizer phrases (CPs) like *that most men sound like...* are fairly common in English, so these language models are presumably expecting these phrases more often than the ones in the grammatical cases like *who most men sound like*. The ungrammatical sentences almost seem like garden path sentences in the sense that they lead a reader (or a language model) to believe they are following along properly. Except, with a garden path sentence, there is a disambiguating point that forces a reader to reevaluate their understanding of the sentence to align with the new information they have encountered. In the ungrammatical examples here, there is no point of disambiguation because one does not have to decide between two interpretations. There is only one interpretation, and it happens to be ungrammatical because of some omitted information.

Test	Grammatical	Ungrammatical
Wh vs *That* with Gap	Nina has learned who most men sound like.	Nina has learned that most men sound like.
Wh vs *That* with Gap Long-Distance	Martin did find out what every cashier that shouldn't drink wore.	Martin did find out that every cashier that shouldn't drink wore.

Table 5.1: Two of the Filler-Gap Dependency tests that all models underperformed on.

Knowing the syntactic parse for these sentences is unlikely to be helpful, as we see in the

figures below. The differences in the parse trees between the grammatical and ungrammatical sentences for each minimal pair are minute. The grammatical sentences that contain the *Wh*-phrases have just one extra non-terminal node compared to the ungrammatical sentences with *that*. Once the RNNG passes the word *who*, the two non-terminals that it sits beneath are immediately combined into one, and then the rest of the sentence structure remains the same. This subtle difference might be difficult to pass from the RNNG to the transformer, so the student model presumably resorts to the construction that is more common (embedded CPs with a *that* complementizer). This may explain why all models scored so poorly on these tests, both in this project, and in the original BLiMP results. If the automatic parser were to register a trace or gap in the *Wh*- sentences, indicating where some information had been omitted at the surface level, then perhaps the RNNG, and thereby the distilled transformer, would be able to register the difference between the two sentences, given that they would have a more distinct structure.

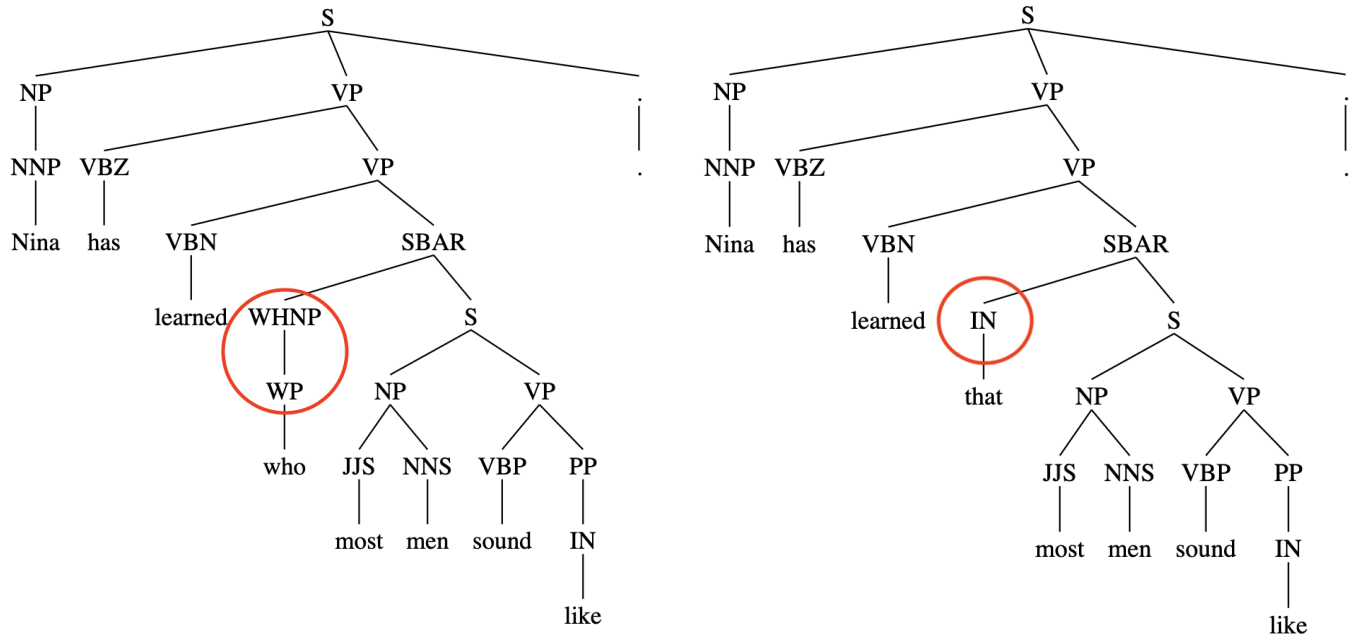

Figure 5.1: Syntax trees for *Wh-* vs. *That* sentences highlighting the subtle differences in structure. The left figure shows the grammatical example, with the corresponding ungrammatical version on the right. Note that the **IN** node label represents subordinating conjunctions/complementizers like *that*, which are paired with entire sentences next to it.

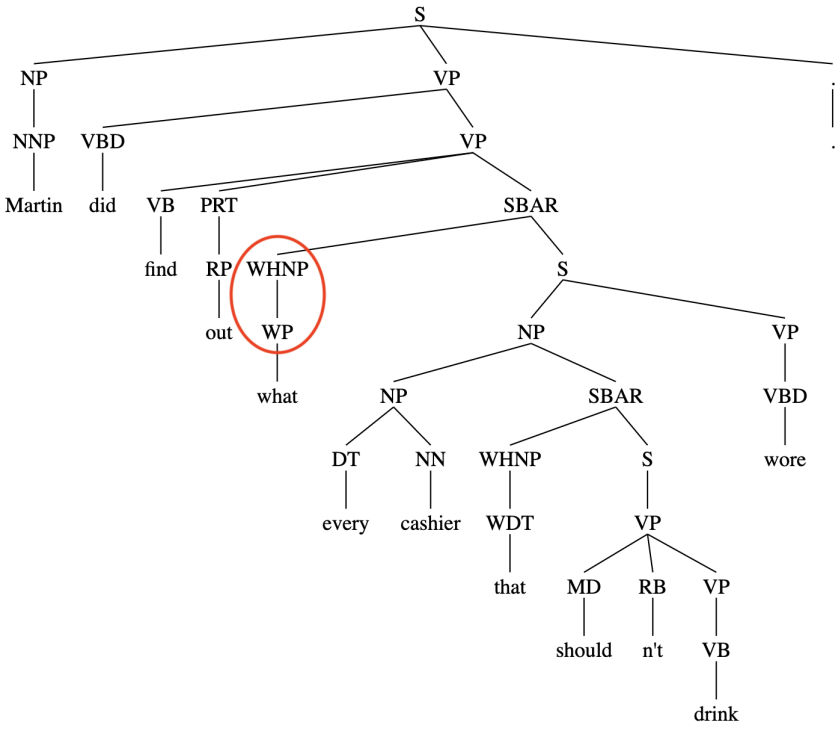

Figure 5.2: Grammatical tree for *Wh-* vs. *That* long distance gap.

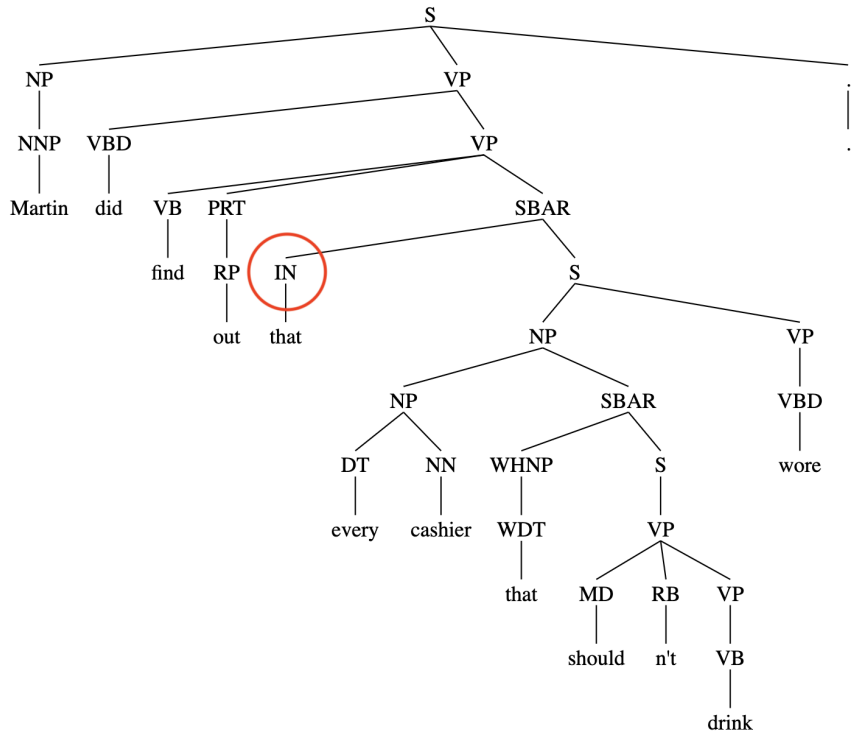

Figure 5.3: Ungrammatical tree for *Wh-* vs. *That* long distance gap.

Chapter 6

Conclusion

In this project, we examined the effects of syntactic signals during language model training. This was done by utilizing knowledge distillation with a recurrent neural network grammar (RNNG) as the teacher, and an autoregressive transformer as the student. These models were evaluated both during training by monitoring their validation perplexity, as well as after training by evaluating their understanding of various linguistic phenomena. This was done using the BLiMP evaluation (Warstadt et al., 2020), where models had to assign probabilities to minimal pairs of grammatical and ungrammatical sentences, with the model being correct when it assigns a higher probability to the grammatical sentence.

The datasets used in this project provided some interesting findings. On the one hand, the BLLIP corpus provides 40M words of well formed, syntactically structured sentences. The consistency of structures across this dataset that accompanies edited written articles allows models to converge quickly. The BabyLM corpus, on the other hand, more closely mimics the type of language one might hear across their lifespan in terms of both content and structure. This diversity posed some challenges for model convergence. Scaling up the model size alone did not seem to help much with this. Increasing the model size along with the aid of a syntactic teacher (a weak one at that) *did* seem to help.

This project attempts to address the formalism versus functionalism split in linguistics as well. A functionalist might look to the fact that the increase in data alone brought about a 9-10% increase in overall BLiMP accuracy. In parallel, a rapid decline in PPL as

the amount of training words was incrementally increased was also observed. This could indicate that the grammatical patterns within language can be learned implicitly through more exposure. While we do observe this phenomenon, it cannot be ignored that the aid of a syntactic teacher certainly helped. These two seemingly opposing views need not be mutually exclusive though. The functionalist account for language does not entirely discard syntax. Syntactic parsing still could be a plausible explanation for the word order in a language, which is supported by recent research in computational and neurolinguistics (Hale et al., 2018).

Moreover, knowledge distillation with a syntactic teacher might be especially effective for training language models on languages with limited textual data. In each case of training models on 5M words or less, the syntactically distilled models converged on a lower perplexity much faster than their control counterparts. This was done with an alpha value of 0.6, meaning only 40% of the loss function came from the KL divergence between the student and the teacher. For these small models and small datasets, a smaller alpha proved to be more effective for distillation convergence, seeing as more of the loss would be coming from the RNNG teacher at that point. We decided on 0.6 for all reported models because, after scaling up the model size and datasets substantially, putting more weight in the distillation term of the loss function proved less effective. For data scarce training, though, more emphasis can be placed on the teacher.

§ 6.1 Limitations

With all of this in mind, there are still several limitations with this project. The first is the use of an automatic parser. We used an off the shelf implementation of the Berkeley neural parser from spaCy (Kitaev and Klein, 2018). The efficiency of using an automatic parser comes at the cost of potentially introducing inaccurate parses. We tried to eliminate these both before and after the parsing process as best as we could. However, given that some of

the sub-corpora in the BabyLM corpus do not perfectly adhere to English syntactic rules, alternative parsers may need to be used or built. An alternative consideration could be to extend corpora like the Penn Treebank. This dataset contains a little more than 1M words of human-parsed English sentences. Since the PPLs declined rapidly as the number of training words increased, a Penn Treebank-like corpus containing 2.5M-5M words would serve well as it would provide gold standard parses at an amount of words that a model could truly learn from.

Due to time constraints and limited computing resources, an exhaustive hyperparameter search was also not performed beyond the small PTB dataset. This meant that we used the same alpha value during knowledge distillation for BLLIP and BabyLM, which turns the loss function for the transformer model into a weighted sum between the NLL loss with the ground truth targets and the Kullback-Leibler divergence with the teacher RNNG model. Along with this, we also used the same temperature value of 2.0 for scaling the logits of the teacher model during distillation. A higher temperature value might even out the probability distribution even more, allowing the student model to better capture the similarities between tokens. The flip side of this, however, is that the teacher model's probability distribution can become too watered down, which is especially a problem if the teacher is already making predictions with low confidence.

This leads into the other major limitation of this work. In order to capture the distribution of words/tokens in a corpus like the BabyLM one, we would need a sufficient model that can properly encode a representation of that corpus. RNNGs are notoriously cumbersome to train (Kuncoro et al., 2019), and large recurrent neural networks in general are difficult to train as they get larger due to vanishing/exploding gradients. This means that scaling the RNNG teacher model up so that it can provide better signals to the transformers has a ceiling to it. The RNNG used in all experiments only contained 2 hidden layers. We found that when scaling beyond that number, the RNNG tended to underperform. Future research

could include the use of a transformer grammar (Sartran et al., 2022) as the teacher model, seeing as transformers can model longer sequences better. Other model architectures such as Mamba/S6 (Gu and Dao, 2024) would also serve as interesting teachers and student models as well.

Bibliography

Adger, D. (2003). *Core Syntax: A Minimalist Approach*. Oxford University Press, Oxford.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

BNC Consortium, P. (2007). The british national corpus, xml edition.

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.

Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. volume 2006, pages 535–541.

Cavell, S. (1979). *The Claim of Reason: Wittgenstein, Skepticism, Morality, and Tragedy*. Oxford University Press, Oxford.

Cavell, S. (1989). This new yet unapproachable america: Lectures after emerson after wittgenstein.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

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 Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Chomsky, N. (1957). *Syntactic Structures*. Mouton and Co., The Hague.

De Deyne, S., Navarro, D., Perfors, A., Brysbaert, M., and Storms, G. (2018). The "small world of words" english word association norms for over 12,000 cue words. *Behavior Research Methods*, 51.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *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.

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.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016). Recurrent neural network grammars. In Knight, K., Nenkova, A., and Rambow, O., editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

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

Gerlach, M. and Font-Clos, F. (2018). A standardized project gutenberg corpus for statistical analysis of natural language and quantitative linguistics.

Gilkerson, J., Richards, J. A., Warren, S. F., Montgomery, J. K., Greenwood, C. R., Oller, D. K., Hansen, J. H. L., and Paul, T. D. (2017). Mapping the early language environment using all-day recordings and automated analysis. *American Journal of Speech-Language Pathology*, 26(2):248–265.

Gu, A. and Dao, T. (2024). Mamba: Linear-time sequence modeling with selective state spaces.

Hale, J., Dyer, C., Kuncoro, A., and Brennan, J. (2018). Finding syntax in human encephalography with beam search. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2727–2736, Melbourne, Australia. Association for Computational Linguistics.

Harris, Z. (1954). Distributional structure. 10:146–162.

Hendrycks, D. and Gimpel, K. (2023). Gaussian error linear units (gelus).

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network.

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

Kim, Y., Rush, A. M., Yu, L., Kuncoro, A., Dyer, C., and Melis, G. (2019). Unsupervised recurrent neural network grammars.

Kitaev, N. and Klein, D. (2018). Constituency parsing with a self-attentive encoder. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Kuncoro, A., Dyer, C., Rimell, L., Clark, S., and Blunsom, P. (2019). Scalable syntax-aware language models using knowledge distillation. In Korhonen, A., Traum, D., and Màrquez, L., editors, *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.

Lenci, A. (2018). Distributional models of word meaning. *Annual Review of Linguistics*, 4(Volume 4, 2018):151–171.

Levshina, N. (2022). *Communicative Efficiency: Language Structure and Use*. Cambridge University Press.

Lison, P. and Tiedemann, J. (2016). OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929. European Language Resources Association (ELRA).

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization.

Luccioni, A. S., Viguier, S., and Ligozat, A.-L. (2022). Estimating the carbon footprint of bloom, a 176b parameter language model.

Macwhinney, B. (2000). The childes project: tools for analyzing talk. *Child Language Teaching and Therapy*, 8.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Maslej, N., Fattorini, L., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Ngo, H., Niebles, J. C., Parli, V., Shoham, Y., Wald, R., Clark, J., and Perrault, R. (2023). Artificial intelligence index report 2023.

McClosky, D., Charniak, E., and Johnson, M. (2008). Bllip north american news text, general release ldc2008t14. Web Download.

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

Millière, R. and Buckner, C. (2024). A philosophical introduction to language models – part i: Continuity with classic debates.

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.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sag, I., Wasow, T., and Bender, E. (2003). *Syntactic Theory: A Formal Introduction*. CSLI lecture notes. Center for the Study of Language and Information.

Samuel, D., Kutuzov, A., Øvrelid, L., and Velldal, E. (2023). Trained on 100 million words and still in shape: BERT meets British National Corpus. In Vlachos, A. and Augenstein, I., editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1954–1974, Dubrovnik, Croatia. Association for Computational Linguistics.

Sartran, L., Barrett, S., Kuncoro, A., Stanojević, M., Blunsom, P., and Dyer, C. (2022). Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. *Transactions of the Association for Computational Linguistics*, 10:1423–1439.

Searle, J. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–57.

Shazeer, N. (2020). Glu variants improve transformer.

Shleifer, S., Weston, J., and Ott, M. (2021). Normformer: Improved transformer pretraining with extra normalization.

Sportiche, D., Koopman, H., and Stabler, E. (2013). *An Introduction to Syntactic Analysis and Theory*. Wiley.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C., and Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–374.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.

Thomas, M. (2020). *Formalism and Functionalism in Linguistics: The Engineer and the Collector*. Routledge, 1st edition.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). Training data-efficient image transformers & distillation through attention.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023a). Llama: Open and efficient foundation language models.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023b). Llama 2: Open foundation and fine-tuned chat models.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Warstadt, A., Mueller, A., Choshen, L., Wilcox, E., Zhuang, C., Ciro, J., Mosquera, R., Paranjabe, B., Williams, A., Linzen, T., and Cotterell, R. (2023). Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In Warstadt, A., Mueller, A., Choshen, L., Wilcox, E., Zhuang, C., Ciro, J., Mosquera, R., Paranjabe, B., Williams, A., Linzen, T., and Cotterell, R., editors, *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.

Warstadt, A., Parrish, A., Liu, H., Mohananey, A., Peng, W., Wang, S.-F., and Bowman, S. R. (2020). BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Wikimedia Foundation, P. (2003). Simple english wikipedia. Software, E-Resource. Retrieved from the Library of Congress: https://www.loc.gov/item/2019205402/.

Wittgenstein, L. (1953). *Philosophical Investigations*. Basil Blackwell, Oxford.

Zhang, Y., Warstadt, A., Li, X., and Bowman, S. R. (2021). When do you need billions of words of pretraining data? In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online. Association for Computational Linguistics.

Appendix A

Linguistic Phenomenon	**Specific Test**	**Control BLLIP**	**Distilled BLLIP**	**Control BabyLM 48**	**Distilled BabyLM 48**	**Control BabyLM 72**	**Distilled BabyLM 72**
Anaphor Agreement	Anaphor Gender Agreement	54.2	60.0	94.1	97.1	93.8	97.6
	Anaphor Number Agreement	80.5	84.9	98.0	97.6	97.8	99.1
Argument Structure	Inchoative	51.1	50.2	48.1	51.2	49.2	51.2
	Animate Subject Passive	70.4	67.7	72.6	75.4	72.1	77.2
	Animate Subject Trans	73.8	70.5	85.9	81.1	82.3	85.5
	Causative	58.4	57.5	68.7	65.3	67.4	69.3
	Passive 2	61.0	62.2	78.1	76.3	75.3	81.2
	Passive 1	64.0	63.6	83.0	83.8	81.4	83.3
	Intransitive	62.8	63.5	48.8	52.5	59.9	62.3
	Drop Argument	69.6	68.2	49.2	53.9	57.7	59.6
	Transitive	65.2	64.0	78.9	78.8	78.7	78.1
Binding	Principle A Domain 2	57.4	58.9	82.8	78.4	84.6	79.4
	Principle A Domain 1	82.5	85.0	99.9	99.5	96.1	99.7
	Principle A C Command	60.9	61.1	65.7	64.2	56.4	59.2
	Principle A case 1	95.8	97.3	100.0	100.0	99.9	99.9
	Principle A case 2	64.9	71.2	90.0	89.6	90.7	87.9
	Principle A Reconstruction	64.1	72.6	22.8	19.4	19.2	22.2
	Principle A Domain 3	48.1	50.1	65.0	62.7	68.0	67.0
Control Raising	Existential There Object Raising	62.3	64.6	67.8	65.9	76.0	67.6
	Existential There Subject Raising	76.5	75.9	85.2	83.2	84.6	83.1
	Expletive It Object Raising	61.0	62.1	69.4	65.6	69.3	70.2
	Tough vs Raising 2	79.0	78.7	85.7	80.1	88.7	85.5
	Tough vs Raising 1	31.2	31.0	51.2	49.8	44.0	50.0
Det.-Noun Agreement	Determiner Noun Agreement with Adj Irregular 1	68.9	72.8	80.7	81.3	79.0	79.7

Linguistic Phenomenon	**Specific Test**	**Control BLLIP**	**Distilled BLLIP**	**Control BabyLM 48**	**Distilled BabyLM 48**	**Control BabyLM 72**	**Distilled BabyLM 72**
	Determiner Noun Agreement with Adj 2	76.8	76.5	85.9	89.1	87.4	88.6
	Determiner Noun Agreement Irregular 2	76.5	78.4	86.3	90.0	82.0	90.2
	Determiner Noun Agreement Irregular 1	68.1	70.6	81.3	79.7	83.8	80.7
	Determiner Noun Agreement 2	82.4	85.6	89.4	92.9	86.1	93.7
	Determiner Noun Agreement 1	81.3	85.3	90.2	91.4	89.2	93.6
	Determiner Noun Agreement with Adjective 1	77.6	80.3	89.2	88.3	84.3	88.7
	Determiner Noun Agreement with Adj Irregular 2	77.0	74.3	84.2	88.2	84.1	86.1
Ellipsis	Ellipsis N-bar 2	49.4	52.7	70.5	59.2	61.3	64.0
	Ellipsis N-Bar 1	49.2	54.8	68.9	73.5	58.8	69.3
Filler Gap Dependency	*Wh* vs *That* with Gap	23.8	22.4	15.4	22.2	22.9	28.9
	Wh-Questions Object Gap	63.9	66.7	86.6	86.4	84.4	88.3
	Wh-Questions Subject Gap	62.9	60.1	92.9	93.2	94.5	94.4
	Wh-Questions Subject Gap Long-Distance	81.8	77.2	88.9	89.0	87.9	87.0
	Wh vs *That* No Gap	90.1	87.3	98.7	96.8	98.3	98.8
	Wh vs *That* No Gap Long-Distance	86.8	88.0	98.4	98.2	97.3	98.8
	Wh vs *That* with Gap Long-Distance	16.5	15.9	5.7	9.2	12.0	11.3
Irregular Forms	Irregular Past Participle Adjectives	74.7	71.0	87.0	78.5	76.0	93.0
	Irregular Past Participle Verbs	81.8	77.3	89.4	87.4	87.9	90.8
Island Effects	Sentential Subject Island	52.1	46.6	27.7	35.1	34.7	37.4

Linguistic Phenomenon	**Specific Test**	**Control BLLIP**	**Distilled BLLIP**	**Control BabyLM 48**	**Distilled BabyLM 48**	**Control BabyLM 72**	**Distilled BabyLM 72**
	Wh-Island	50.7	58.5	75.4	63.1	69.9	76.8
	Adjunct Island	57.9	64.1	83.3	81.1	81.0	84.4
	Left Branch Island Echo Question	48.9	42.9	71.6	61.9	70.3	58.8
	Complex NP Island	43.7	50.1	53.0	50.1	51.8	50.6
	Coordinate Structure Constraint Complex Left Branch	25.9	31.8	63.6	52.7	59.3	57.7
	Coordinate Structure Constraint Object Extraction	73.9	72.8	79.3	77.8	82.9	82.3
	Left Branch Island Simple Question	31.3	36.1	77.4	65.8	74.9	73.2
NPI Licensing	Only NPI Scope	56.8	53.0	75.5	82.3	74.3	72.6
	Sentential Negation NPI Scope	42.9	43.9	58.1	62.4	62.2	61.1
	Only NPI Licensor Present	68.1	71.8	67.4	86.6	78.1	90.4
	Matrix Question NPI Licensor Present	93.9	93.9	99.2	99.8	99.9	99.9
	NPI Present 1	60.1	59.7	45.4	56.50	51.3	63.6
	NPI Present 2	63.6	61.7	58.6	55.0	57.5	68.3
	Sentential Negation NPI Licensor Present	93.0	96.3	97.4	98.8	99.5	98.5
Quantifiers	Superlative Quantifiers 1	80.8	82.6	93.6	87.0	92.7	66.2
	Existential *There* Quantifiers 2	16.6	19.2	41.9	43.0	28.9	31.2
	Existential *There* Quantifiers 1	85.1	85.4	95.8	97.7	99.4	99.3
	Superlative Quantifiers 2	76.4	71.7	87.7	82.9	75.9	83.2
Subject Verb Agreement	Regular Plural Subject Verb Agreement 2	77.6	73.6	79.8	75.8	79.9	84.5
	Regular Plural Subject Verb Agreement 1	73.8	78.8	87.1	85.6	85.3	80.1

Linguistic Phenomenon	**Specific Test**	**Control BLLIP**	**Distilled BLLIP**	**Control BabyLM 48**	**Distilled BabyLM 48**	**Control BabyLM 72**	**Distilled BabyLM 72**
	Irregular Plural Subject Verb Agreement 2	74.2	75.8	82.2	75.6	83.8	86.6
	Irregular Plural Subject Verb Agreement 1	64.8	66.5	82.4	75.6	81.5	80.9
	Distractor Agreement Relative Clause	55.9	56.8	63.4	69.6	64.5	68.9
	Distractor Agreement Relational Noun	68.7	66.1	72.7	78.2	73.3	79.8
Average		64.5	65.3	74.6	74.1	74.1	75.8