Mind the Gap: How BabyLMs Learn Filler-Gap Dependencies

Anonymous ACL submission

Abstract

Humans acquire syntactic constructions like filler–gap dependencies from limited and often noisy input. Can neural language models do the same? We investigate this question by evaluating GPT-2 models trained on child-directed input from the BabyLM Challenge. Our experiments focus on whether these "baby" language models acquire filler–gap dependencies, generalize across constructions, and respect structural constraints such as island effects. We apply a suite of syntactic constructions to four models trained on child language, including two base models (trained on 10M and 100M tokens) and two well-performing models from the BabyLM Challenge (ConcreteGPT and BabbleGPT). We evaluate model behavior using wh-licensing scores, flip tests, and grammaticality contrasts across four constructions. Results show that BabyLM-scale models partially acquire filler–gap dependencies but often fail to generalize or fully capture island constraints.

1 Introduction

Babies are remarkable learners, but how they do so remains a central question in language acquisition research (Yang, 2016). A long-standing debate concerns whether the linguistic input children receive is sufficient to explain the grammatical knowledge they develop. According to the Poverty of the Stimulus (POS) argument, children's input is too sparse and underspecified to support acquisition of certain abstract structures, thus innate learning biases are required (Chomsky, 1968, 1973, 1980; Pearl, 2022). However, this remains an open debate, with some arguing that domain-general, input-driven learning suffices (Lewis and Elman, 2001), while others argue that domain-specific innate knowledge is necessary (Yang, 2004).

A particularly relevant test case for this debate involves structural dependencies like *filler-gap* constructions. These involve a *filler* (e.g., a wh-phrase such as *who* and *what*) that **licenses** a *gap* – an empty syntactic position often spanning intervening structure (e.g., *Who did the intern say _ left the meeting early?*). Such dependencies are constrained by certain syntactic structures – for example, they can be blocked by certain syntactic islands (e.g., **What did he leave [before she finished _]*$_{\text{island}}$*?*) (Ross, 1967; Huang, 1982). Substantial research shows that human learners acquire these patterns during early childhood with limited input exposure (Omaki et al., 2015; Gagliardi et al., 2016; Atkinson et al., 2018; Perkins and Lidz, 2021).

Prior work has shown that RNN-based language models trained on vast amounts of data can demonstrate sensitivity to these dependencies and even some island effects across multiple languages, though their generalizations often remain shallow and construction-specific (Wilcox et al., 2018; Bhattacharya and van Schijndel, 2020; Chaves, 2020; Ozaki et al., 2022; Kobzeva et al., 2023; Howitt et al., 2024). Meanwhile, recent studies have begun comparing neural language models (LMs) to human language acquisition on basic syntactic patterns that require hierarchical representation of the sentence structures, such as subject–verb agreement and yes/no question formation (Yedetore et al., 2023; Evanson et al., 2023). What remains less understood is whether LMs can acquire more complex, non-local generalizations like filler–gap dependencies from small datasets that are developmentally plausible, especially in contexts that involve structural constraints such as islands.

Lan et al. (2024) supports the POS argument by showing that models trained on limited data fail to learn complex long-distance dependencies like parasitic gaps, likely due to insufficient input richness, suggesting a role for innate biases in human learning. However, their focus on rare constructions leaves open whether such models can generalize to core filler–gap dependencies and show human-like

sensitivity to structural constraints such as island effects. This motivates the following research questions addressed in our study:

1. Can LMs trained on child-directed input acquire filler-gap dependencies?
2. Do they exhibit human-like sensitivity to structural constraints, such as island effects?
3. What do their successes and failures tell us about the nature of linguistic generalization?

To answer these questions, we build a suite of syntactic evaluations inspired by prior work (Wilcox et al., 2018; Ozaki et al., 2022; Howitt et al., 2024), covering four constructions that test key aspects of filler-gap knowledge: gap distance, multiple gaps, and two types of islands. We apply these to four GPT-2 models trained on datasets from the BabyLM challenge (Warstadt et al., 2023, `https://babylm.github.io/index.html`): two base models (trained on 10M and 100M tokens), and two competitive models from the BabyLM Challenge (henceforth **BabyLM models**), ConcreteGPT (Capone et al., 2024) and BabbleGPT (Goriely et al., 2024).[1] A vanilla pre-trained GPT-2 serves as a high-resource benchmark. Our key findings are as follows:

- No model, including the vanilla pre-trained GPT-2, captures the full structural generalizations consistently across constructions.
- All BabyLM models (10M or 100M words) show partial acquisition of filler-gap dependencies. Models trained on 100M tokens outperform 10M-token models.
- GPT-2-100M and BabbleGPT learn gap-distance dependencies and some island effects (especially for wh-islands), but fail to generalize consistently across constructions.
- ConcreteGPT, despite its smaller scale, shows evidence of partially capturing the bijectivity of filler-gap dependencies for several constructions.

2 Methodology

This project investigates the acquisition of filler-gap dependencies by GPT models trained on child-language data. Our methodology builds upon established work assessing syntactic generalization in LMs (Wilcox et al., 2018; Ozaki et al., 2022; Howitt et al., 2024).

2.1 "Baby" Language Models

To approximate the limited and sparse linguistic input of early human language acquisition, we trained GPT-2-small models (Radford et al., 2019) on datasets from the BabyLM challenge (Warstadt et al., 2023), which consist of 10M and 100M English words, with a large proportion of child and child-directed language (Charpentier et al., 2025, `https://babylm.github.io/`). These models serve as base models to assess the performance of a standard GPT-2 architecture trained on developmentally plausible amounts of data. Notably, these models were trained using standard training procedures without specialized techniques, reflecting the limited and unstructured input characteristic of early language acquisition in children.

In addition to these base models, we include two well-performing GPT-2 models from the 2024 BabyLM challenge (Hu et al., 2024): ConcreteGPT (Capone et al., 2024) and BabbleGPT (Goriely et al., 2024). ConcreteGPT, trained on the 10M-word dataset, incorporates a curriculum learning approach. By utilizing concreteness ratings from Brysbaert et al. (2014), training data was ordered to introduce simpler, more concrete language patterns before progressing to more abstract structures, to mirror the developmental trajectory of human language acquisition. BabbleGPT represents one of the most advanced models trained on the BabyLM 100M-word dataset. An innovative input transformation approach of converting text data into phoneme streams was applied, simulating the early stages of human language acquisition where children process spoken language before written text.

The inclusion of both the base GPT-2 models and the competitive BabyLM models of similar architecture provides a valuable comparison of how training data size and learning strategies influence model performance on syntactic tasks. While the base models offer insight into the general behavior of LMs trained on child-language data, the BabyLM models bring in specialized optimization approaches. These models represent some of the best results achievable with constrained child-language data and offer a clear benchmark for understanding the potential and limitations of typ-

[1] We focus on standard decoder-only architectures to maintain consistency in our comparisons, and abstract away from hybrid models such as GPT-BERT (Charpentier et al., 2025).

ical LMs – GPT-2 specifically – in this context. Finally, we also include an unconstrained GPT-2 model, pre-trained on a 40GB corpus, as the high-performance upper bound.[2]

All models and tokenizers were trained from scratch using Hugging Face's GPT-2 implementation (Wolf et al., 2020), adhering to established model specifications (Sennrich et al., 2015; Radford et al., 2019; Brown et al., 2020). See Appendix A for details on training configurations.

2.2 Experimental Design

We adopt a 2×2 factorial design following Wilcox et al. (2018) and Wilcox et al. (2023) to test whether GPT-style language models, when trained with child-directed speech data, are capable of acquiring filler-gap dependencies. In particular, we manipulate the presence of a wh-licensor (i.e., a filler) and the presence of a syntactic gap in each sentence.

The following shows the basic filler-gap licensing conditions (Wilcox et al., 2018):

1. [−FILLER, −GAP]: I know that the lion devoured a gazelle at sunrise.
2. [+FILLER, −GAP]: *I know what the lion devoured a gazelle at sunrise.
3. [−FILLER, +GAP]: *I know that the lion devoured [___] at sunrise.
4. [+FILLER, +GAP]: I know what the lion devoured [___] at sunrise.

Based on the basic filler-gap licensing conditions, we designed a suite of syntactic evaluation items to probe whether language models generalize filler-gap dependencies across distinct constructions. Following Wilcox et al. (2018, 2023), we investigate four of the most-studied syntactic constructions known to influence how filler-gap dependencies are processed by humans (Ross, 1967; Huang, 1982; Wilcox et al., 2018, 2023): gap distance, double gaps, wh-islands, and adjunct islands. Each construction includes 20 items. This resulted in 5 sub-datasets: `gap-distance-obj` and `gap-distance-PP` for gap distance sentences with different gap positions, `double-gaps`, `wh-islands`, and `adjunct-islands`. In particular, we focus on **structural factors** like filler-gap distance length, presence of multiple gaps, and island constraints. Full details of the testing materials are provided in Appendix B.

[2]We use the GPT-2 model hosted on Hugging Face under the MIT License and comply with its terms of use and intended use.

3 Evaluation Metrics

The primary evaluation metric is **surprisal**, calculated as: $S_t = -\log_2 P(w_t \mid w_1, w_2, \ldots, w_{t-1})$ where w_t is the target word at position t, and the probability is conditioned on the preceding context $w_1, w_2, \ldots, w_{t-1}$. We calculate surprisals at critical regions of the sentences. This includes **local surprisal** measuring the post-gap region, and **global surprisal** assessing effects across the whole embedded clause (Ozaki et al., 2022). Wilcox et al. (2018) consistently measured local surprisal in the region after the potential gap. Following Wilcox et al. (2019b); Ozaki et al. (2022), we adopt a different practice to account for surprisal spikes of illicitly filled gaps that occur at the filled gap region. For [-gap] sentences, local suprisal is measured at the filled gap position, while for [+gap] sentences it is measured at the post-gap region. Additionally, global suprisal is normalized by clause length to control for possible confounding effects as [+gap] sentences tend to be shorter, resulting in lower total suprisals (Ozaki et al., 2022). These metrics are measured and tested to examine the co-occurrence expectations of fillers and gaps more exhaustively.

We use the **wh-licensing score** (aka. wh-licensing interaction, licensing interaction, and filler-gap interaction, Wilcox et al., 2018) to measure the degree to which the presence of a wh-licensor reduces the surprisal at the gap position:

$$[S(\text{+filler, -gap}) - S(\text{-filler, -gap})] - [S(\text{+filler, +gap}) - S(\text{-filler, +gap})]$$

where S stands for surprisal. This compares the surprisal values across the four combinations of [+filler/-filler] and [+gap/-gap] conditions. An idealized wh-licensing score would show a large positive difference in the [-gap] condition and a large negative difference in the [+gap] condition, suggesting that the model expects a gap when a filler is present and penalizes unlicensed gaps.

Following Wilcox et al.'s (2018) experimental setup, we fit mixed-effects linear regression models on filler-gap conditions to predict the two surprisal metrics mentioned above, including random intercepts by sentence sets. This is to determine by statistical significance whether the model has correctly acquired the rules and constraints surrounding filler-gap dependencies. The fixed effect

structure includes filler-gap conditions and structural conditions (e.g., filler-gap distance for the gap distance construction, gap count for the double gap construction) for basic construction types, and filler-gap conditions and island types for island constructions. For double gap constructions, since potential gap positions can be at either the subject position or the object position, or both, we do not measure local surprisal for this construction as the target regions for measurement are inconsistent.

We adopt two additional tests proposed by Ozaki et al. (2022): the **flip test** and the **grammaticality division test**. The flip test requires that the surprisal difference (between [+filler] and [–filler] conditions) flips its direction depending on the presence of a gap. That is, the filler should reduce surprisal in the [+gap] condition (i.e., the presence of the filler helps reduce the uncertainty at the gap), but increase surprisal in the [–gap] condition (i.e., the filler increases uncertainty when there is no gap to license). It specifically tests for the bijectivity of filler-gap dependencies. The grammaticality division test, on the other hand, directly compares the surprisal of grammatical sentences with that of their ungrammatical counterparts, to see whether the model assigns lower surprisal to grammatical configurations.

Both of these tests are performed through mixed-effects linear regression modeling. In the flip tests, surprisal is predicted on [filler] for [+gap] and [-gap] sentences separately; in the grammaticality division test, we assign a grammaticality variable [gram] to all sentences and predict surprisal on [gram], while treating [+gram] as the baseline. For basic constructions, we assign [gram] depending on whether the numbers of fillers and gaps satisfy the one-to-one relationship of filler-gap dependencies; for island constructions, the presence of islands blocks the filler-gap dependency, rendering [+filler, +gap] sentences ungrammatical. The grammaticality division test would only be conducted on global surprisal, since the location where the local surprisal is measured in a sentence now confounds the presence of gaps (Ozaki et al., 2022).

4 Primary Results

We evaluate the models' learning of filler-gap dependencies through calculating wh-licensing scores, and determining statistical significance through mixed effects modeling tests. All results reported are statistically significant ($p < 0.05$) unless stated otherwise.

4.1 Licensing-Gap Interaction

As our threshold model, GPT-2 demonstrates expected licensing behavior in most of the constructions. Learning of the dependency is observed with `gap-distance-obj` and `gap-distance-PP`, both locally and globally. However, treating gap distance length as a continuous variable, we see that gap distance length poses negative effects on wh-licensing score for `gap-distance-obj`, when measuring global surprisal ($\beta = 0.003$, $p < 0.01$). This indicates that intervening material length affects the model's judgment of filler-gap licensing relationships, with longer intervening material rendering a gap more surprising even with the presence of a filler. Although exhibiting similar patterns of decreasing wh-licensing scores as length increases in other conditions, as seen in Figure 1 and Figure 2, the model remains statistically robust to intervening material length for all other metrics and gap positions in this construction. Measuring global surprisal for `double-gaps`, we discover that when a filler is present, GPT-2 finds the absence of a corresponding gap more surprising than licensed single gaps, although the model does not find double gaps more surprising in a statistically significant manner ($p = 0.553$).

Results also show that GPT-2 has learned island constraints to a certain extent. When island constraints are present, wh-licensing scores are expected to decrease. With `wh-islands`, island constraints lead to lower wh-licensing scores when compared to the non-island baseline, in all conditions. As for `adjunct-islands`, we see reduction in the wh-licensing score of the adjunct-back condition when compared to the object condition baseline, in both post-gap and embedded clause regions. We also see reduction in the wh-licensing score of the adjunct front condition when compared to the object condition baseline, but results are only significant when measured in the embedded clause region (local: $p = 0.06$, global: $p < 0.01$).

The trained models in general show evidence for partial filler-gap dependency representation, possibly due to limitations of the training corpora sizes. GPT-2-10M does not learn the filler-gap dependency with statistical significance at all, for any of the constructions. GPT-2-100M acquires the dependency for `gap-distance-obj` both locally and globally, while staying robust to different lengths

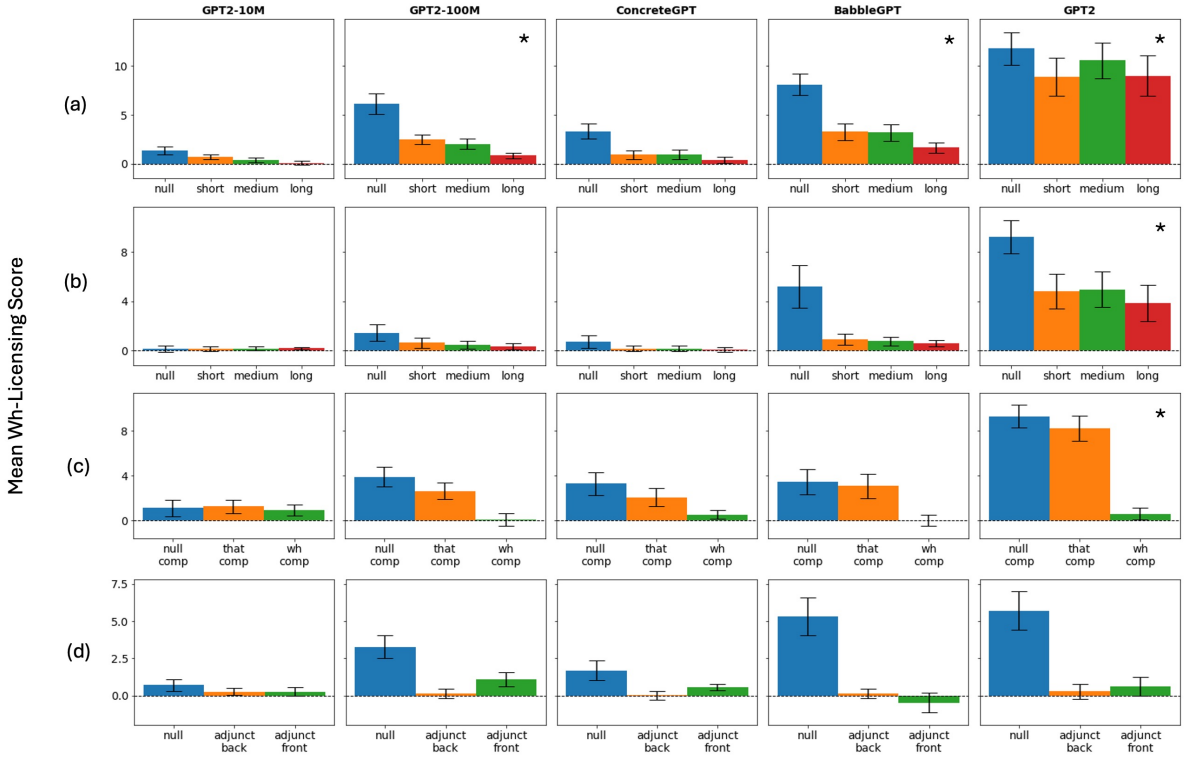

Figure 1: Wh-Licensing Score with Local Surprisals. Each row represents the results for one construction: (a) gap-distance-obj, (b) gap-distance-pp, (c) wh-islands, (d) adjunct-islands. Constructions fully learned with statistical significance (robust to intervening factors and capturing island constraints) are marked by asterisks.

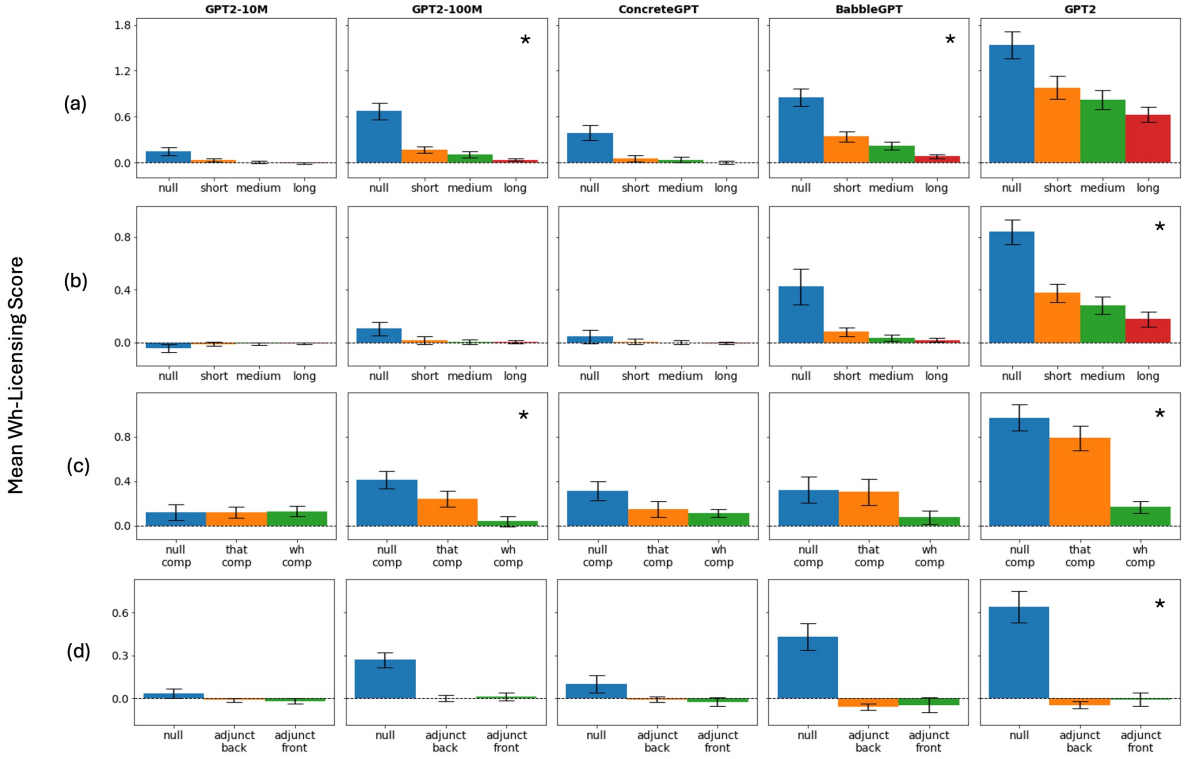

Figure 2: Wh-Licensing Score with Global Surprisals. Each row represents the results for one construction: (a) gap-distance-obj, (b) gap-distance-pp, (c) wh-islands, (d) adjunct-islands. Constructions fully learned with statistical significance (robust to intervening factors and capturing island constraints) are marked by asterisks.

of intervening material. For double-gaps, GPT-2-100M exhibits licensing behavior for global surprisals with licensed single gaps being less surprising than the lack of a gap when given a filler. The presence of illicit double gaps does not pose statistically significant effects on surprisal values ($p = 0.817$). However, through directly plotting out surprisals, we see that mean surprisal increases with the number of illicit gaps for both the GPT-2-10M and GPT-2-100M models (Figure 3). For wh-islands, GPT-2-100M is shown to have acquired the filler-gap dependency alongside with island constraints globally. It also shows usual licensing behavior for global adjunct-islands, however failing to recognize island constraints.

Comparing the two BabyLM models with our base models, we see slight improvement in filler-gap dependency capabilities on the 10M scale. While GPT-2-10M shows no statistically significant evidence in acquiring any of the constructions, its 10M-word counterpart ConcreteGPT displays usual licensing behavior for global wh-islands ($p < 0.05$) but ignores island constraints. Similar to GPT-2-100M, BabbleGPT acquires gap-distance-obj (local: $p < 0.01$, global: $p < 0.01$) but not gap-distance-PP. With island constructions, it displays usual licensing behavior globally for wh-islands and both locally and globally for adjunct-islands, again completely overlooking island constraints.

4.2 Flips

In flip tests, positive signs of correct licensing behavior would be for the presence of a filler to render a sentence more surprising with [-gap] sentences, and less surprising with [+gap] sentences. In the case of island constraints, island sentences should be viewed as more surprising under the [+filler, +gap] condition, when compared to non-island baselines.

The threshold GPT-2 model shows good performance in flip tests in general. The GPT-2 model passes the flip test for gap-distance-obj and gap-distance-PP with local surprisal. This holds true when measuring global surprisal for gap-distance-obj, but does not hold true for gap-distance-PP. With double-gaps, when given a filler, the model finds gapless sentences more surprising, which is the expected behavior. In the [+gap] direction, it however does not find licensed gaps less surprising, or illicit double gaps more surprising. In the case of wh-islands, the GPT-2 model passes the flip test in both the post-gap and embedded clause regions. Under the [+filler, +gap] condition, island sentences are found to be more surprising than baseline non-island sentences, showing that the model is aware of island constraints. Results with adjunct-islands are rather mixed, with the model failing to capture the [+gap] direction of the bijectivity globally. When considering island constraints for [+filler, +gap] sentences, it finds the adjunct-back condition more surprising than the non-island baseline, while in the adjunct-front condition islandhood does not pose any significant effects on local surprisal ($p = 0.057$).

For the trained models, we can see that while the GPT-2-10M did not fully acquire filler-gap dependencies, it does capture half of the filler-gap bijectivity for some of the constructions. GPT-2-10M partially captures the [-gap] direction of the filler-gap bijectivity for gap-distance-obj, where the presence of a filler increases local surprisal, but not global surprisal ($p = 0.052$). The model also exhibits usual flipping behavior for wh-islands with local surprisal, and captures the [+gap] direction of the bijectivity with global surprisal, however showing no acknowledgments of island constraints in both cases. Similarly, for adjunct-islands, it captures the [-gap] direction of the bijectivity with local surprisals, while islandhood does not pose any significant effect on surprisals. GPT-2-100M passes the flip test for local gap-distance-obj and local wh-islands, correctly rendering islandhood as more surprising. It also showcases usual flipping behavior for global wh-islands and local adjunct-islands, however failing to show effective islandhood effects. It captures half of the bijectivity for global gap-distance-obj ([gap-]), local gap-distance-PP ([gap+]), and global adjunct-islands ([gap-]), while somehow capturing islandhood effects for the adjunct-front condition (β = 0.14, $p < 0.05$). For double-gaps, both of the models find gapless sentences more surprising when given a filler.

ConcreteGPT captures one direction of the filler-gap bijectivity for the majority of the constructions globally, showing improvement over its 10M-word trained counterpart GPT-2-10M. It captures the [-gap] direction for global gap-distance-obj, global gap-distance-PP, double-gaps, local and global adjunct-islands, and the [+gap]

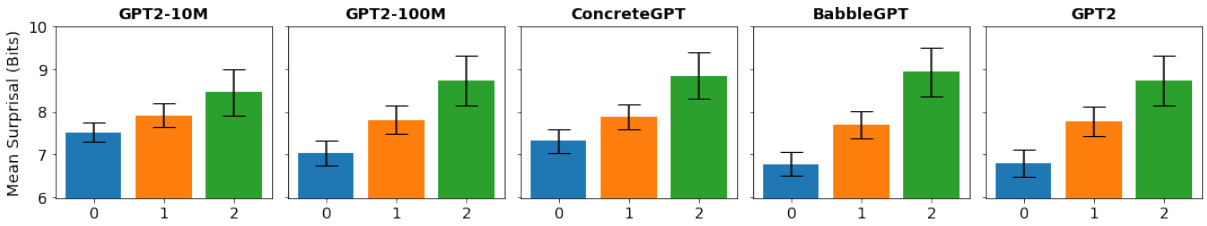

Figure 3: Double Gaps Mean Surprisal (y-axis) as a function of Number of Illicit Gaps (x-axis)

Construction	GPT-2	GPT-2-10M	GPT-2-100M	ConcreteGPT	BabbleGPT
gap_distance_obj	0.497***	0.024 (p = 0.771)	0.124 (p = 0.08)	0.061 (p = 0.481)	0.187**
gap_distance_pp	0.209**	-0.010 (p = 0.898)	0.016 (p = 0.823)	0.005 (p = 0.948)	0.069 (p = 0.337)
double_gaps	1.015***	0.389***	0.806***	0.647***	1.017***
wh_islands	0.64***	0.218**	0.323***	0.191**	0.366***
adjunct_islands	0.513***	0.131 (p = 0.093)	0.271***	0.179*	0.489***

Table 1: Estimated effect of grammaticality on surprisal. Baseline is [+gram]. A positive value denotes increase in surprisal when a sentence is ungrammatical, which is the expected behavior. Significance levels: *$p < .05$, **$p < .01$, ***$p < .001$.

direction for global `wh-islands`. It demonstrates flips for `wh-islands` locally, though islandhood does not render a sentence more surprising under the [+filler, +gap] condition, as it should have. Similarly, BabbleGPT captures the [-gap] direction of the bijectivity for several constructions, including local and global `gap-distance-obj`, global `gap-distance-PP`, `double-gaps`, global `wh-islands`, and global `adjunct-islands`. It passes the flip test for `wh-islands` and `adjunct-islands` locally, while recognizing island constraints. Full details of the flip test results can be found in Appendix C.

4.3 Division by Grammaticality

The GPT-2 model passes the grammaticality test for all constructions, all with high statistical significance, as seen in Table 1. In contrast, our trained models do not perform as well as the threshold model on the grammaticality test. GPT-2-10M passes the grammaticality test for `double-gaps` and `wh-islands`. GPT-2-100M passes the grammaticality test for `double-gaps`, `wh-islands`, and `adjunct-islands`.

The BabyLM models show stronger abilities in judging grammaticality. ConcreteGPT passes the grammaticality test for `double-gaps`, `wh-islands`, and `adjunct-islands`. BabbleGPT passes the grammaticality test for all constructions except for `gap-distance-PP`, displaying competency nearing the threshold GPT-2 model in judging grammaticality.

5 Model Retraining on Enriched Corpus

The POS argument posits that children, despite limited exposure to direct evidence, are capable of acquiring complex linguistic structures. Our training corpora contained many instances of the adjunct island construction (10M: 4,145; 100M: 20,115),[3] yet both trained models failed to learn it. This alone suggests that purely statistical learners lack the innate biases that support human language acquisition.

Following Lan et al. (2024), we conducted an additional experiment to investigate whether enhanced exposure to a specific construction could improve the model's learning of that construction. We augmented the training data with additional instances of adjunct island sentences, a construction which both of our trained models failed to acquire, and retrained the models. Models were retrained using the exact same training configurations, with additional adjunct island instances that are separate from the testing suite. If enhanced exposure leads to better performance, this would suggest that the models' failures stem from a lack of sufficient statistical evidence, highlighting the role of data volume in model learning. If the models still fail to learn the construction after the data augmentation, it would further emphasize that the issue is not simply one of insufficient exposure, but rather a divergence between the learning mechanisms of statistical learners and human learners.

[3] Instances were identified using the spaCy package.

Despite the additional training materials, neither of the retrained models fully learned the adjunct island construction. The retrained GPT-2-100M shows usual licensing behavior for global `adjunct-islands` but fails to recognize island constraints, similar as before. Though looking at flip test results we do see slight improvement. GPT-2-100M passes the flip test for local `adjunct-islands` ($p < 0.01$), while recognizing both the adjunct-front and adjunct-back island conditions that it failed to recognize before ($p < 0.05$).

Increased exposure to the adjunct island construction did lead to slightly better performance, however, the models still fail to fully capture the construction. This highlights a fundamental difference between statistical learners and human learners: when given language input of similar scale, current models are not comparable to human language learners, supporting the argument that models lack the innate learning biases that humans use to acquire complex linguistic structures.

6 Discussion and Conclusion

This study investigates the ability of language models trained on child-directed speech data to learn and generalize filler-gap dependencies, specifically on how these models handle basic constructions and complex structures such as island constraints.

Our results show that while models trained on the BabyLM corpus exhibit limited success in fully acquiring filler-gap dependencies, GPT-2-small models gradually learn the licensing relationship when trained on a larger corpus. While GPT-2-10M fails to learn the full filler-gap relationship for any constructions with statistical significance, the model shows evidence of learning half of the filler-gap bijectivity for several constructions, as demonstrated by the flip test results. GPT-2-100M fully learns the dependency for gap distance and global wh-islands, however failing to learn island constraints for adjunct islands.

Moreover, we demonstrate that models often struggle with generalizing across gap positions under increased distance. In particular, even the threshold GPT-2 model shows reduced wh-licensing scores with longer intervening material in several conditions, suggesting that long-distance dependencies remain a challenge. This pattern aligns with findings from child language acquisition, where such dependencies are known to be acquired relatively late (Atkinson et al., 2018).

While the threshold GPT-2 model generally displays correct behavior in assessing island constraints, flip test results show that it does not necessarily capture the full bijectivity of filler-gap dependencies. In particular, the model's performance on island constructions remains mixed, suggesting that while the model can identify island constraints, it does not fully learn the intricate dependencies between fillers and gaps when island constraints are involved. Our results support previous work, which suggests that even if a computational model is able to approximate human acceptability judgments, inductive biases are necessary to reliably acquire island constraints (Pearl and Sprouse, 2013). This is in opposition to what naturally occurs in human learners, where these patterns are acquired in early childhood with limited input exposure and gives merit to the idea that humans have innate mechanisms which aid language acquisition (Gagliardi et al., 2016; Atkinson et al., 2018).

BabyLM models show stronger performance on filler-gap dependencies and grammaticality judgments than base models at the 10M scale, while results at the 100M scale remain mixed. At 10M, flip test results indicate that although ConcreteGPT fails to fully acquire filler-gap dependencies, it captures half of the bijectivity in more constructions than GPT-2-10M. At 100M, both models learn the object gap condition in the gap distance construction, but only GPT-2-100M shows sensitivity to island constraints in global wh-islands, which BabbleGPT fails to achieve. BabyLM models also outperform our trained models in grammaticality judgments. These findings suggest that while specialized training techniques may yield modest gains in filler-gap learning, complex constraints like islands remain difficult.

The enriched corpus experiment reveals that while models benefit from additional training materials, they still do not reach human-level language capabilities even with ample exposure to language.

To summarize, when trained on child-like language input, the examined language models fail to exhibit the structure-sensitive generalizations that characterize human language acquisition, particularly in filler-gap dependencies. This discrepancy between model and human learning offers empirical support for a central claim of the POS argument that some aspects of syntactic knowledge may not be learnable from input alone and likely require additional inductive biases.

7 Limitations

While our study focuses on whether child-language-trained models can acquire filler-gap dependencies, several limitations constrain the generalizability of our findings.

First, all models in this study share the same underlying architecture – GPT-2 – differing only in terms of training data volume and optimization strategies. While this consistency was maintained to enable controlled comparisons, it also limits the scope of our conclusions. Larger models or architectures with different inductive biases, such as other decoder-only transformers like LLaMA or hybrid models like GPT-BERT, may exhibit fundamentally different behaviors in learning and generalizing syntactic dependencies. Furthermore, while we performed hyperparameter tuning to optimize model performance, the search space we explored was limited due to computational constraints, and it is possible that alternative hyperparameter configurations might have yielded better syntactic generalization.

In addition, the training data used in this study, although drawn from the BabyLM corpora and designed to reflect developmentally plausible language input, remains limited in linguistic diversity. These corpora represent only a narrow slice of the kinds of input that English-spoken children encounter during language acquisition. They lack exposure to multimodal grounding, prosody, and certain rare or edge-case syntactic constructions. This narrow linguistic bandwidth may hinder the models' ability to fully acquire complex grammatical phenomena, particularly those involving abstract or less frequent dependencies such as island constraints.

Taken together, these limitations caution against broad generalizations from our results and underscore the need for further research across diverse model architectures, training regimes, and linguistic inputs.

References

Emily Atkinson, Matthew W Wagers, Jeffrey Lidz, Colin Phillips, and Akira Omaki. 2018. Developing incrementality in filler-gap dependency processing. *Cognition*, 179:132–149.

Debasmita Bhattacharya and Marten van Schijndel. 2020. Filler-gaps that neural networks fail to generalize. In *Proceedings of the 24th conference on computational natural language learning*, pages 486–495.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46:904–911.

Luca Capone, Alessandro Bondielli, and Alessandro Lenci. 2024. ConcreteGPT: A baby GPT-2 based on lexical concreteness and curriculum learning. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 189–196, Miami, FL, USA. Association for Computational Linguistics.

Lucas Charpentier, Leshem Choshen, Ryan Cotterell, Mustafa Omer Gul, Michael Hu, Jaap Jumelet, Tal Linzen, Jing Liu, Aaron Mueller, Candace Ross, and 1 others. 2025. Babylm turns 3: Call for papers for the 2025 babylm workshop. *arXiv preprint arXiv:2502.10645*.

Rui P Chaves. 2020. What don\'t rnn language models learn about filler-gap dependencies? *Society for Computation in Linguistics*, 3(1).

Noam Chomsky. 1968. *Language and Mind*. Cambridge University Press.

Noam Chomsky. 1973. Problems of knowledge and freedom. *Philosophy*, 48(184).

Noam Chomsky. 1980. On cognitive structures and their development: A reply to piaget. *Language and learning: the debate between Jean Piaget and Noam Chomsky*, pages 35–52.

Linnea Evanson, Yair Lakretz, and Jean-Rémi King. 2023. Language acquisition: do children and language models follow similar learning stages? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12205–12218.

Annie Gagliardi, Tara M Mease, and Jeffrey Lidz. 2016. Discontinuous development in the acquisition of filler-gap dependencies: Evidence from 15-and 20-month-olds. *Language Acquisition*, 23:234–260.

Zébulon Goriely, Richard Diehl Martinez, Andrew Caines, Paula Buttery, and Lisa Beinborn. 2024. From babble to words: Pre-training language models on continuous streams of phonemes. In *The 2nd*

BabyLM Challenge at the 28th Conference on Computational Natural Language Learning, pages 37–53, Miami, FL, USA. Association for Computational Linguistics.

Kevin Howitt, Sunil Nair, Amelia Dods, and Robert M Hopkins. 2024. Generalizations across filler-gap dependencies in neural language models. *Proceedings of the 28th Conference on Computational Natural Language Learning*.

Michael Y. Hu, Aaron Mueller, Candace Ross, Adina Williams, Tal Linzen, Chengxu Zhuang, Ryan Cotterell, Leshem Choshen, Alex Warstadt, and Ethan Gotlieb Wilcox. 2024. Findings of the second BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning*, pages 1–21, Miami, FL, USA. Association for Computational Linguistics.

C.-T. James Huang. 1982. Logical relations in chinese and the theory of grammar. *MIT Working Papers in Linguistics*, 18:1–288.

Anastasia Kobzeva, Suhas Arehalli, Tal Linzen, and Dave Kush. 2023. Neural networks can learn patterns of island-insensitivity in norwegian. In *Proceedings of the Society for Computation in Linguistics*, pages 175–185.

Nur Lan, Emmanuel Chemla, and Roni Katzir. 2024. Large language models and the argument from the poverty of the stimulus. *Linguistic Inquiry*, pages 1–28.

John D Lewis and Jeffrey L Elman. 2001. Learnability and the statistical structure of language: Poverty of stimulus arguments revisited. In *Proceedings of the 26th annual Boston University conference on language development*, volume 1, pages 359–370. Citeseer.

Akira Omaki, Ellen F Lau, Imogen Davidson White, Melissa L Dakan, Amanda Apple, and Colin Phillips. 2015. Hyper-active gap filling. *Frontiers in Psychology*, 6:384.

Satoru Ozaki, Daniel Yurovsky, and Lauren Levin. 2022. How well do lstm language models learn filler-gap dependencies? In *Proceedings of the Society for Computation in Linguistics 2022*, pages 76–88.

Lisa Pearl. 2022. Poverty of the stimulus without tears. *Language Learning and Development*, 18(4):415–454.

Lisa Pearl and Jon Sprouse. 2013. Syntactic islands and learning biases: Combining experimental syntax and computational modeling to investigate the language acquisition problem. *Language Acquisition*, 20(1):23–68.

Laurel Perkins and Jeffrey Lidz. 2021. Eighteen-month-old infants represent nonlocal syntactic dependencies. *Proceedings of the National Academy of Sciences*, 118(41):e2026469118.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

John Robert Ross. 1967. *Constraints on Variables in Syntax*. Phd thesis, Massachusetts Institute of Technology.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.

Ethan Wilcox, Richard Futrell, and Roger Levy. 2023. Using computational models to test syntactic learnability. *Linguistic Inquiry*, early access:1–44.

Ethan Wilcox, Roger Levy, and Richard Futrell. 2019b. What syntactic structures block dependencies in rnn language models? *Proceedings of the Annual Meeting of the Cognitive Science Society*, 41.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do rnn language models learn about filler-gap dependencies? *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Charles Yang. 2016. *The Price of Linguistic Productivity: How Children Learn to Break the Rules of Language*. MIT Press.

Charles D Yang. 2004. Universal grammar, statistics or both? *Trends in cognitive sciences*, 8(10):451–456.

Aditya Yedetore, Tal Linzen, Robert Frank, and R. Thomas McCoy. 2023. How poor is the stimulus? evaluating hierarchical generalization in neural networks trained on child-directed speech. In

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9370–9393, Toronto, Canada. Association for Computational Linguistics.

A Model Training

A.1 Data Preprocessing

Our preprocessing approach follows the methodology for handling CHILDES data outlined by Yedetore et al. (2023), with additional modifications to better suit our needs. The process includes format standardization converting all text into plain text format through applying the NLTK (Bird and Loper, 2004)[4] CHILDESCorpusReader for XML parsing, punctuation and spacing normalization preserving contractions (don't → do n't) to align with CHILDES Treebank tokenization standards, non-linguistic content filtering removing annotation markers (e.g., [laughter], [noise]) and extra non-linguistic characters (e.g. placeholder token "xxx"), and child-directed speech filtering retaining only child-directed utterances.

In addition to the CHILDES corpus, the BabyLM dataset also consists of multiple smaller datasets from the Gutenburg Project, Open Subtitles, Simple Wikipedia, and Switchboard corpora. Preprocessing for these datasets focused on removing non-linguistic features such as added headers, special characters outside of those used for punctuation, and line-by-line labels which showed speakers in the Open Subtitle dataset.

We allocated 90% of the final corpus to training and 10% to validation.

A.2 Tokenizer Training

The process of training a tokenizer from scratch is a crucial step in preparing data for language model training. We employed Byte Pair Encoding (BPE) tokenizers compatible with GPT models. Below are the detailed configurations and steps involved in the tokenizer training process.

1. Training Corpus: We used the same child-language input data corpora we used for model training to train the tokenizer. This is good practice as it ensures consistency in tokenization and vocabulary alignment between the tokenizer and the model. The GPT-2-10M tokenizer was trained on the 10M-word dataset, while the GPT-2-100M tokenizer was trained on the 100M-word dataset, meticulously controlling for training data sizes comparable to child-language input.

2. Tokenizer Initialization: BPE tokenizers were initialized using the HuggingFace tokenizers library. The tokenizers were then configured with the several key components listed below.

   - Normalizer: This component ensures that the text is cleaned and normalized before tokenization. We applied multiple normalization steps, including:
     - Prepend: Prepending spaces for byte-level tokenization to ensure format consistency.
     - NFKC: Normalization form KC (Compatibility Composition) for Unicode normalization.
     - Replace: We applied regular expression-based replacements to handle newline characters and extra spaces.
   - Pre-tokenizer: This component breaks the input text into smaller parts before the BPE algorithm is applied. In this case, we used a Whitespace pre-tokenizer to split text on spaces.
   - Decoder: The decoder reverses the tokenization process. Here, we used a sequence of decoding steps that handle byte-level decoding, and stripped spaces.
   - Post-processor: This step is responsible for adding special markers such as the start-of-sequence token. In this case, we configured the post-processor for byte-level token processing without trimming offsets.

3. Tokenizer Training: The BPE tokenizers were trained on the prepared corpus files with a vocabulary size of 50,257 tokens. We set a minimum frequency of 2 for the inclusion of tokens in the vocabulary. No special tokens were defined during training.

4. Saving the Tokenizer: After training, the tokenizers were saved as JSON format files to later be used for tokenization during model training and evaluation.

[4] https://www.nltk.org/

A.3 Model Training

Once the tokenizer was trained, the next step was to train the GPT-2 model using the prepared tokenizer. Below are the steps and configurations used for the model training process.

1. Model Configuration: The GPT-2 model configuration was set up using the GPT-2Config class from the HuggingFace Transformers library (`https://huggingface.co/docs/transformers/en/model_doc/gpt2`). The model's configuration was aligned with the specifications typically used for GPT-2-small.

2. Dataset Loading: The training datasets were tokenized using the respective trained BPE tokenizers. This was to ensure that the data was encoded in the appropriate format that the GPT-2 model could process.

3. Training Arguments: The training hyperparameters were specified using the TrainingArguments class, which defines how the model would be trained. Key hyperparameters include:

    - Training Epochs: 10
    - Batch Size: A batch size of 1 was chosen to fit GPT-2's large size within the available GPU memory.
    - Gradient Accumulation: To simulate a larger effective batch size while conserving memory, gradient accumulation was employed, with the `gradient_accumulation_steps` parameter set to 16. This means that gradients were accumulated over 16 steps before an update to the model's parameters occurred.
    - Learning Rate: The learning rate was set to 5e-4. This rate was found to provide a balance between performing effective training and avoiding issues related to overshooting the optimal weights.
    - Warmup Ratio: A warmup ratio of 0.1 was applied, meaning that 10% of the total training steps were used for the gradual warmup of the learning rate. This helps stabilize training in the early stages by avoiding large gradient updates.
    - Weight Decay: Weight decay was applied at a rate of 0.01, a typical value for regularization, to help prevent overfitting by penalizing large model weights.
    - Learning Rate Scheduler: A cosine learning rate scheduler was used. This scheduler reduces the learning rate in a cosine manner, starting high and gradually decaying to zero, which is effective for ensuring stable convergence towards the end of training.
    - Adam Optimizer: The Adam optimizer was used with default beta values: `adam_beta1 = 0.9` and `adam_beta2 = 0.999`. These values help control the momentum and moving averages of the gradient during optimization.
    - Precision: To optimize computational efficiency, mixed-precision training was enabled using `fp16 = True`, which allows the model to use 16-bit floating-point precision instead of 32-bit precision, reducing memory usage and speeding up computation without significant loss in accuracy.

4. Trainer Initialization: The Trainer class from HuggingFace was used to manage the training loop, including data loading and model updates. The trainer was initialized with the model, the tokenizer, and training arguments. The model training process included the use of EarlyStoppingCallback with the patience parameter set to 3. This callback monitors the validation loss and halts training if no improvement is observed for three consecutive evaluation steps, helping to prevent overfitting and unnecessary computation.

Training and validation losses were logged every 100 steps. See Figure 4 for the loss curves of GPT-2-10M and GPT-2-100M. Training rounds took roughly 4 GPU hours per round for 10M models, and 50 GPU hours per round for 100M models using V100 double precision GPUs.

B Testing Materials

We designed a suite of syntactic evaluation items to probe whether language models generalize filler-gap dependencies. Following Wilcox et al. (2018, 2023), we focus on four of the most-studied syntactic constructions known to influence how filler-gap dependencies are processed by humans (Ross,

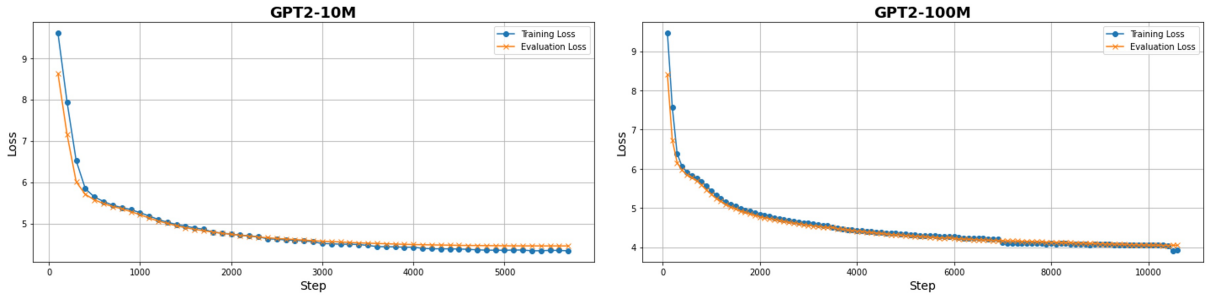

Figure 4: Training and Evaluation Loss Curves of the Trained GPT-2 Models.

1967; Huang, 1982). Each construction includes 20 items.

Gap Distance Here we test how increasing the amount of intervening material (in the form of relative clause modifiers) affects the model's ability to maintain long-distance dependencies. This condition is split into two subparts: direct object gaps and indirect object gaps. Each modifier is varied in length.

Gap Distance with DO Gap

(2) a. The manager predicts what the intern forwarded [___] to the client earlier this morning. [+FILLER, +GAP, NO MODIFIER]

b. The manager predicts what the intern who you admire forwarded [___] to the client earlier this morning. [+FILLER, +GAP, SHORT MODIFIER]

c. The manager predicts what the intern who you worked closely with on the project forwarded [___] to the client earlier this morning. [+FILLER, +GAP, MEDIUM MODIFIER]

d. The manager predicts what the intern who you recommended highly after the summer project forwarded [___] to the client earlier this morning. [+FILLER, +GAP, LONG MODIFIER]

Gap Distance with IO Gap

(3) a. The manager predicts who the intern forwarded an important email to [___] earlier this morning. [+FILLER, +GAP, NO MODIFIER]

b. The manager predicts who the intern who you admire forwarded an important email to [___] earlier this morning. [+FILLER, +GAP, SHORT MODIFIER]

c. The manager predicts who the intern who you worked closely with on the project forwarded an important email to [___] earlier this morning. [+FILLER, +GAP, MEDIUM MODIFIER]

d. The manager predicts who the intern who you recommended highly after the summer project forwarded an important email to [___] earlier this morning. [+FILLER, +GAP, LONG MODIFIER]

Multiple Gaps This condition tests whether the model can handle the presence of more than one gap in the same clause. We include subject gaps, object gaps, and double gap sentences.

(4) a. James realized what [___] chased the cat through the yard. [+FILLER, +GAP, SUBJECT GAP ONLY]

b. James realized what the dog chased [___] through the yard. [+FILLER, +GAP, OBJECT GAP ONLY]

c. *James realized what [___] chased [___] through the yard. [+FILLER, +GAP, SUBJECT AND OBJECT GAPS]

Island Constraints To evaluate whether models can learn syntactic constraints on long-distance dependencies, we include two classic island types, following those proposed in Ross (1967) and further formalized in Huang (1982). Specifically, we test the model's sensitivity to wh-islands and adjunct islands. These constructions are known to block filler–gap dependencies in adult grammars and are considered central to the study of structural locality in syntax. We do not include sentential subject islands, as their status in child acquisition of filler–gap dependencies remains unclear and warrants further empirical confirmation.

Wh-Islands This condition tests whether the model suppresses filler–gap expectations when the gap is embedded in a syntactic wh-island (e.g., a whether-clause). The complementizer of the embedded clause is varied (null, that, whether), following the design in Wilcox et al. (2018).

(5) a. The teacher discovered what the student claimed his friend lost [___] during the field trip. [+FILLER, +GAP, NULL-COMPLEMENTIZER]

b. The teacher discovered what the student claimed that his friend lost [___] during the field trip. [+FILLER, +GAP, THAT-COMPLEMENTIZER]

c. The teacher discovered what the student claimed whether his friend lost [___] during the field trip. [+FILLER, +GAP, WH-COMPLEMENTIZER]

Adjunct Islands In this condition, the gap is embedded in an adjunct clause introduced by "while." We test three versions: no adjunct, adjunct attached at the back, and fronted adjunct, following the design in Wilcox et al. (2018).

(6) a. We discovered what the intern at the new office was preparing for [___] with extra care. [+FILLER, +GAP, NO ADJUNCT]

b. We discovered what the lights went out while the intern at the new office was preparing for [___] with extra care. [+FILLER, +GAP, ADJUNCT BACK]

c. We discovered what while the intern at the new office was preparing for [___] with extra care the lights went out. [+FILLER, +GAP, ADJUNCT FRONT]

C Flip Test Results

We include full details of flip test results in Table 2.

		GPT-2		GPT-2-10M		GPT-2-100M		ConcreteGPT		BabbleGPT	
		local	global	local	global	local	global	local	global	local	global
gap_distance_obj	[gap-]	2.599***	0.258***	0.33***	0.095 (p = 0.052)	0.861***	0.162***	0.595 (p = 0.274)	0.166**	1.211**	0.198***
	[gap+]	-2.424***	-0.239***	0.021 (p = 0.817)	0.072 (p = 0.061)	-0.578***	0.038 (p = 0.486)	-0.103 (p = 0.855)	0.106(p = 0.1)	-0.811 (p = 0.104)	0.011 (p = 0.83)
gap_distance_pp	[gap-]	0.772**	0.183***	-0.009 (p = 0.838)	0.046 (p = 0.347)	0.136 (p = 0.074)	0.06 (p = 0.186)	0.072 (p = 0.883)	0.113*	0.252 (p = 0.52)	0.128**
	[gap+]	-2.083***	-0.026 (p = 0.633)	-0.093 (p = 0.211)	0.056 (p = 0.307)	-0.226*	0.04 (p = 0.332)	-0.073 (p = 0.902)	0.108*	-0.68 (p = 0.237)	0.059 (p = 0.195)
double_gaps	[gap = 0]		0.039***		0.136***		0.245***		0.156***		0.439***
	[gap = 1]		-0.088 (p = 1.44)		0.08 (p = 0.089)		-0.077 (p = 0.134)		-0.086 (p = 0.191)		-0.067 (p = 0.279)
	[gap = 2]		-0.024 (p = 0.725)		0.059 (p = 0.149)		-0.078 (p = 0.102)		-2.43***		-0.189**
wh_islands	[gap-]	2.236***	0.255***	0.297**	-0.02 (p = 0.415)	0.928***	0.087**	1.12***	0.043 (p = 0.267)	1.083***	0.202***
	[gap+]	-2.404***	-0.23***	-0.254*	-0.08**	-1.014***	-0.119***	-0.514**	-0.112*	-0.63***	0.041 (p = 0.456)
	islandhood (wh_comp)	2.127***	0.157**	0.035 (p = 0.808)	-0.002 (p = 0.966)	0.955***	0.031 (p = 0.487)	0.413 (p = 0.09)	-0.049 (p = 0.466)	0.589*	-0.116 (p = 0.136)
adjunct_islands	[gap-]	2.043***	0.341***	0.297***	0.08 (p = 0.07)	1.229***	0.222***	1.05***	0.294***	1.706***	0.36***
	[gap+]	-0.825**	0.022 (p = 0.571)	-0.049 (p = 0.66)	0.062 (p = 0.198)	-0.41*	0.088*	0.203 (p = 0.179)	0.243***	-0.905***	0.145**
	islandhood (adjunct_front)	0.632 (p = 0.057)	0.203***	0.076 (p = 0.631)	0.06 (p = 0.381)	0.195 (p = 0.447)	0.14*	-0.151 (p = 0.48)	0.002 (p = 0.978)	0.99**	0.115 (p = 0.129)
	islandhood (adjunct_back)	0.665*	0.186**	-0.011 (0.944)	0.016 (p = 0.819)	0.437 (p = 0.088)	-0.009 (p = 0.873)	-0.07 (p = 0.742)	-0.168**	0.898**	0.103 (p = 0.176)

Table 2: Flip test results. Listed are the estimated effects when [+filler]. With the presence of a filler, [-gap] should see an increase in surprisal (positive value), while [+gap] should see a decrease in surprisal (negative value). Islandhood effects are estimated under [+filler, +gap]. When a filler-gap relationship exists given island constraints, there should be an increase in surprisal (positive value).