Structural generalization is hard for sequence-to-sequence models

Yuekun Yao and **Alexander Koller**
Department of Language Science and Technology
Saarland Informatics Campus
Saarland University, Saarbrücken, Germany
{ykyao, koller}@coli.uni-saarland.de

Abstract

Sequence-to-sequence (seq2seq) models have been successful across many NLP tasks, including ones that require predicting linguistic structure. However, recent work on compositional generalization has shown that seq2seq models achieve very low accuracy in generalizing to linguistic structures that were not seen in training. We present new evidence that this is a general limitation of seq2seq models that is present not just in semantic parsing, but also in syntactic parsing and in text-to-text tasks, and that this limitation can often be overcome by neurosymbolic models that have linguistic knowledge built in. We further report on some experiments that give initial answers on the reasons for these limitations.

1 Introduction

Humans are able to understand and produce linguistic structures they have never observed before (Chomsky, 1957; Fodor and Pylyshyn, 1988; Fodor and Lepore, 2002). From limited, finite observations, they generalize at an early age to an infinite variety of novel structures using recursion. They can also assign meaning to these, using the Principle of Compositionality. This ability to generalize to unseen structures is important for NLP systems in low-resource settings, such as underresourced languages or projects with a limited annotation budget, where a user can easily use structures that had no annotations in training.

Over the past few years, large pretrained sequence-to-sequence (seq2seq) models, such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), have brought tremendous progress to many NLP tasks. This includes linguistically complex tasks such as broad-coverage semantic parsing, where e.g. a lightly modified BART set a new state of the art on AMR parsing (Bevilacqua et al., 2021). However, there have been some concerns that seq2seq models may have difficulties with *compositional generalization*, a class of tasks in semantic parsing where the training data is structurally impoverished in comparison to the test data (Lake and Baroni, 2018; Keysers et al., 2020). We focus on the COGS dataset of Kim and Linzen (2020) because some of its generalization types specifically target *structural generalization*, i.e. the ability to generalize to unseen structures.

In this paper, we make two contributions. First, we offer evidence that structural generalization is systematically hard for seq2seq models. On the semantic parsing task of COGS, seq2seq models don't fail on compositional generalization as a whole, but specifically on the three COGS generalization types that require generalizing to unseen linguistic structures, achieving accuracies below 10%. This is true both for BART and T5 and for seq2seq models that were specifically developed for COGS. What's more, BART and T5 fail similarly on syntax and even POS tagging variants of COGS (introduced in this paper), indicating that they do not only struggle with *compositional* generalization in semantics, but with *structural* generalization more generally. Structure-aware models, such as the compositional semantic parsers of Liu et al. (2021) and Weißenhorn et al. (2022) and the Neural Berkeley Parser (Kitaev and Klein, 2018), achieve perfect accuracy on these tasks.

Second, we conduct a series of experiments to investigate what makes structural generalization so hard for seq2seq models. It is not because the encoder loses structurally relevant information: One can train a probe to predict COGS syntax from BART encodings, in line with earlier work (Hewitt and Manning, 2019; Tenney et al., 2019a); but the decoder does not learn to use it for structural generalization. We find further that the decoder does not even learn to generalize semantically when the input is enriched with syntactic structure. Finally, it is not merely because the COGS tasks require the mapping of language into symbolic represen-

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 5048 - 5062
December 7-11, 2022

	Training	Generalization
(a) LEX subj_to_obj (common noun)	A hedgehog ate the cake. *cake(x_4); hedgehog(x_1) ∧ eat.agent(x_2, x_1) ∧eat.theme(x_2, x_4)	The baby liked the hedgehog. *baby(x_1); *hedgehog(x_4); like.agent(x_2, x_1) ∧ like.theme(x_2, x_4)
(b) STRUCT PP recursion	Ava saw a ball in a bowl on the table. *table(x_9); see.agent(x_1, Ava) ∧ see.theme(x_1, x_3) ∧ ball(x_3) ∧ ball.nmod.in(x_3, x_6) ∧ bowl(x_6) ∧ bowl.nmod.on(x_6, x_9)	Ava saw a ball in a bowl on the table on the floor. *table(x_9); *floor(x_{12}); see.agent(x_1, Ava) ∧ see.theme(x_1, x_3) ∧ ball(x_3) ∧ ball.nmod.in(x_3, x_6) ∧ bowl(x_6) ∧ bowl.nmod.on(x_6, x_9) ∧ table.nmod.on(x_9, x_{12})
(c) STRUCT obj_to_subj PP	Noah ate the cake on the plate. *cake(x_3); *plate(x_6); eat.agent(x_1, Noah) ∧ eat.theme(x_1, x_3) ∧ cake.nmod.on(x_3, x_6)	The cake on the table burned. *cake(x_1); *table(x_4); cake.nmod.on(x_1, x_4) ∧ burn.theme(x_3, x_1)

Figure 1: Some examples from the COGS dataset. LEX represents lexical generalization and STRUCT denotes structural generalization.

tations. We introduce a new text-to-text variant of COGS called *QA-COGS*, where questions about COGS sentences must be answered in English. We find that T5 performs well on structural generalization with the original COGS sentences, but all models still struggle with a harder text-to-text task involving structural disambiguation.

The code[1] and datasets[2] are available online.

2 Related work

The recent interest in compositional generalization has raised concerns about limitations of seq2seq models. For instance, the SCAN dataset (Lake and Baroni, 2018) requires a model to translate natural-language instructions into symbolic action sequences; it has multiple splits in which the test data contains new combinations of commands or instructions that are systematically longer than in training. The PCFG dataset (Hupkes et al., 2020) builds upon SCAN and adds instructions with recursive structure. The CFQ dataset (Keysers et al., 2020) maps questions to SPARQL queries, and splits the data according to a measure of compositional complexity (MCD). In all of these papers, simple seq2seq models based on LSTMs and transformers were shown to perform poorly when the test data was more complex than the training data.

Since then, followup research has shown that both generic transformer-based models (Ontanon et al., 2022; Csordás et al., 2021), general-purpose pretrained models (Furrer et al., 2020), and seq2seq models that are specialized for the task can achieve higher accuracies than the ones reported in the papers introducing the datasets. Nonetheless, there is a sense that despite the best efforts of the community, pure seq2seq models are hitting a ceiling on compositional generalization tasks.

In this paper, we shed some light on the issue by (a) clarifying that seq2seq models do not struggle with compositional generalization per se, but with *structural* generalization, and (b) demonstrating that this type of generalization remains hard for seq2seq models even after heavy pretraining. This is in contrast to most previous research, which has avoided pretraining and focused on length or MCD as the primary source of difficulty. Our data includes instances where the structure, but not the length differs between training and testing, and therefore allows us to differentiate between the two. The importance of structure to compositional generalization is also recognized by Bogin et al. (2022).

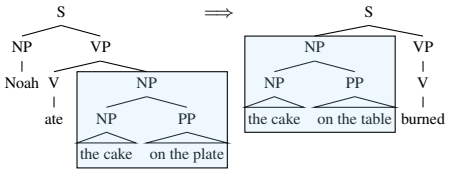
The difficulty of structural generalization for neural models has also been studied in more targeted ways. For instance, Yu et al. (2019) show empirically that LSTM-based seq2seq models cannot learn to close the brackets of Dyck languages, and Hahn (2020) proves that transformers cannot learn to distinguish well-bracketed Dyck expressions. McCoy et al. (2020) find empirically that seq2seq models struggle to learn the structural operations necessary to rewrite declarative English sentences into questions, whereas tree-based models work better.

3 Structural generalization in COGS

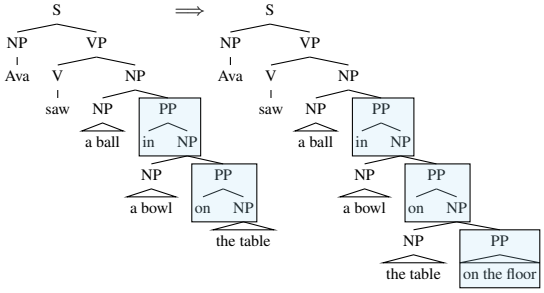
COGS (Kim and Linzen, 2020) is a synthetic semantic parsing dataset in which English sen-

[1] https://github.com/coli-saar/Seq2seq-on-COGS

[2] https://github.com/coli-saar/Syntax-COGS



(a) object PP to subject PP



(b) PP recursion

Figure 2: Structural generalization in COGS.

tences must be mapped to logic-based meaning representations (see Fig. 1 for some examples). It distinguishes 21 *generalization types*, each of which requires generalizing from training instances to test instances in a particular systematic and linguistically-informed way. COGS was designed to measure *compositional generalization*, the ability of a semantic parser to assign correct meaning representations to out-of-distribution sentences. Unlike SCAN and CFQ, it includes generalization types with unbounded recursion and separates them cleanly from other generalization types, both of which are crucial for the experiments reported here.

Most generalization types in COGS are *lexical*: they recombine known grammatical structures with words that were not observed in these particular structures in training. An example is the generalization type "subject to object" (Fig. 1a), in which a noun ("hedgehog") is only seen as a subject in training, whereas it is only used as on object at test time. The syntactic structure at test time was already observed in training; only the words change.

By contrast, *structural generalization* involves generalizing to linguistic structures that were not seen in training (cf. Fig. 1b,c). Examples are the generalization types "PP recursion", where training instances contain prepositional phrases of depth up to two and generalization instances have PPs of depth 3–12; and "object PP to subject PP", where PPs modify only objects in training and only subjects at test time. These structural changes are illustrated in Fig. 2.

Structural generalization requires learning about recursion and compositionality, and is thus a more thorough test of human-like language use, whereas lexical generalization amounts to smart template filling. In this paper, we investigate how well structural generalization can be solved by different classes of model architectures: *seq2seq models* and *structure-aware models*. We define a model as "structure-aware" if it is explicitly designed to encode linguistic knowledge beyond the fact that sentences are sequences of tokens. This captures a large class of models that can be as "deep" as a compositional semantic parser or as "shallow" as a POS tagger that requires that each input token gets exactly one POS tag.

4 Structural generalization is hard for seq2seq

We begin with some evidence that structural generalization in COGS is hard for seq2seq models, while structure-aware models learn it quite easily. We first collect some results on the original semantic parsing task of COGS, extending it with numbers for BART and T5. We then transform COGS into a corpus for syntactic parsing and POS tagging and investigate the ability of BART and T5 to generalize structurally on these tasks.

4.1 Experimental setup: COGS

We follow standard COGS practice and evaluate all models on the generalization set. We report exact match accuracies, averaged across 5 training runs.

Seq2seq models. We train BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) as semantic parsers on COGS. Both models are strong representatives of seq2seq models and perform well across many NLP tasks. To apply these models on COGS, we directly fine-tune the pretrained *bart-base* and *t5-base* model on it with the corresponding tokenizer; see Appendix A for details. We also report results for a wide range of published seq2seq models for COGS (Kim and Linzen, 2020; Conklin et al., 2021; Csordás et al., 2021; Akyürek and Andreas, 2021; Zheng and Lapata, 2022; Qiu et al., 2021).

Structure-aware models. We report evaluation results for LeAR (Liu et al., 2021) and the AM parser (Weißenhorn et al., 2022). Both models learn to predict a tree structure which is decoded into COGS meaning representations using the Principle of Compositionality. Thus both models are structure-aware.

	Model Class	Model	STRUCT Obj to Subj PP	STRUCT CP recursion	STRUCT PP recursion	LEX all 18 other types	Overall
semantics	seq2seq	BART	0	0	12	91	79
		BART+syn	0	5	8	93	80
		T5	0	0	9	97	83
		Kim and Linzen 2020	0	0	0	73	63
		Akyürek and Andreas 2021	0	0	1	96	82
		Zheng and Lapata 2022	0	12	39	99	89
		Conklin et al. 2021	0	0	0	88	75
		Csordás et al. 2021	0	0	0	95	81
		Qiu et al. 2021 *	100	100	100	100	100
	structure-aware	Liu et al. 2021	93	100	99	99	99
		Weißenhorn et al. 2022	78	100	99	100	98
syntax	seq2seq	BART	0	9	22	99	87
		T5	5	7	9	99	86
	structure-aware	Neural Berkeley Parser	84	95	98	100	99
POS tags	seq2seq	BART	0	6	19	98	85
		T5	0	4	4	98	85
	structure-aware	most frequent POS	92	98	100	92	93

Table 1: Exact match accuracies on the individual generalization types. Column LEX reports mean accuracy over the 18 lexical generalization types. *) After structure-aware data augmentation.

4.2 Results

We report the results by generalization type in the "semantic" rows in Table 1. We will explain "BART+syn" in Section 5.3 and the "syntactic" and "POS" sections in Section 4.3.

Structural generalization is hard. We can observe that all recent models achieve near-perfect accuracy on the 18 lexical generalization types. However, all pure seq2seq models achieve very low accuracy on the structural generalization types, whereas structure-aware models are still very accurate. One outlier is the seq2seq model of Qiu et al. (2021). It employs heavy data augmentation based on (structure-aware) synchronous grammars encoding the Principle of Compositionality, which provides training instances of higher recursive depth to the seq2seq model. The seq2seq model then still generalizes to the recursive depth which it has seen in training, but not beyond (Peter Shaw, p.c.).

Note that the mean accuracy is dominated by the lexical generalization types; to really measure the ability of a model to generalize to unseen structures, it is important to focus on the structural generalization types. Note further that BART and T5 perform very well among the class of seq2seq models, outperforming many models that are specialized to COGS. We will focus on these two models in the experiments below.

It is important that although the generalization instances on PP and CP recursion are longer than the training instances, the low accuracy of the seq2seq models cannot be explained exclusively in terms of their known weakness to length generalization (Hupkes et al., 2020). For the "Object to Subject PP" generalization type, the generalization and training sentences have the same length, but different structures. Thus our results point towards a specific weakness to structural generalization.

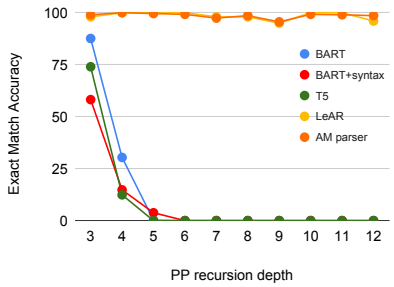


Figure 3: Influence of PP recursion depth on overall PP depth generalization accuracy.

Depth generalization. The accuracy of the seq2seq models depends on the difference in complexity of the test instance and the training data. For instance, all training instances for the "PP recursion" type have recursion depth two or less; Fig. 3 shows how the accuracy depends on the recursion depth of the test instance. As we see, the accuracy of BART (even when informed by syntax, cf. Section 5.3) degrades quickly with recursion depth. By contrast, LeAR and the AM parser maintain high accuracy across all recursion depths.

4.3 Syntax-COGS and POS-COGS

While these results on semantic parsing are suggestive, they could be explained away in many ways. For instance, the weakness of seq2seq models with

respect to structural generalization might be specific to semantics, or the semantic representations chosen in COGS might be idiosyncractic and unfair to seq2seq models.

We therefore investigate structural generalization on a syntax variant of COGS. We convert each training and generalization instance of the COGS corpus into a pair of the sentence with its syntax tree (Syntax-COGS) and a pair of the sentence with its POS tag sequence (POS-COGS). This is possible because COGS is generated from an unambiguous context-free grammar; we reconstruct the unique syntax trees that underly each instance in COGS.

We replace the very fine-grained non-terminals (e.g. `NP_animate_dobj_noPP`) of the original COGS grammar with more general ones (e.g. `NP`) and remove duplicate rules (e.g. `NP`→`NP`) resulting from this. We extract the POS tag sequences from the preterminal nodes of the syntax trees.

We train BART and T5 to predict linearized constituency trees and the POS tag sequences from the input sentences. As a structure-aware model, we use the Neural Berkeley Parser (Kitaev and Klein, 2018), which consists of a self-attention encoder and a chart decoder and therefore has the notion of a tree and its recursive structure built into the parsing model. On the POS tagging task, our "structure-aware" model is constrained to predict exactly one POS tag for each input token. Specifically, we determine the most frequent POS tag in the training data for each word type and assign it to all occurrences of the word during inference.

Results. The results are shown in the "syntactic" and "POS" rows of Table 1. We find the same pattern as in the semantic parsing case: the seq2seq models do well on LEX, but struggle with STRUCT. The structure-aware models handle all generalization types well. Thus, the difficulties that seq2seq models have on structural generalization on COGS are not limited to semantics: rather, they seem to be a general limitation in the ability of seq2seq models to learn linguistic structure from structurally simple examples and use it productively.

We also present an example for *obj_pp_to_subj_pp* type across different tasks in Figure 4. For a sentence *The baby on a tray in the house screamed*, T5 consistently predicted wrong symbol sequences. For example, in semantic parsing, T5 tends to predict *tray* as the theme of *scream* with a PP structure. This might be due to a

	Input	The baby on a tray in the house screamed.
Semantics	Gold	$*\text{baby}(x_1)$; $*\text{house}(x_7)$; $\text{baby.nmod.on}(x_1, x_4)$ ∧ $\text{tray}(x_4)$ ∧ $\text{tray.nmod.in}(x_4, x_7)$ ∧ $\text{scream.agent}(x_8, x_1)$
	T5	$*\text{baby}(x_1)$; $*\text{house}(x_{10})$; $\text{scream.agent}(x_2, x_1)$ ∧ $\text{scream.theme}(x_2, x_4)$ ∧ $\text{tray}(x_4)$ ∧ $\text{tray.nmod.in}(x_4, x_7)$
Syntax	Gold	(S (NP (Det The) (N baby) (PP (P on) (NP (Det a) (N tray) (PP (P in) (NP (Det the) (N house))))))) (VP (V screamed)))
	T5	(S (NP (Det The) (N baby) (VP (V on) (NP (Det a) (N tray) (PP (P in) (NP (Det the) (N house)))))))
POS	Gold	Det N P Det N P Det N V
	T5	Det N V Det N P Det N P Det N

Figure 4: Example for *obj_to_subj_pp* type. We list the annotation of semantic parse, syntax tree and POS tags with corresponding T5 predictions.

preference of T5 to reuse the pattern for object-PP sentences in the train set even if the intransitive verb does not license it. T5 also displays an unawareness of word order that is reminiscent of the difficulties that seq2seq models otherwise face in relating syntax to word order (McCoy et al., 2020). For recursion generalization types, we find that the main error is that the decoder cannot generate long or deep enough sequences.

5 Encoder or decoder?

We now turn to the second question: *Why* do seq2seq models struggle on structural generalization? We start by investigating at which point the model loses the structural information – does the encoder not represent it, or can the decoder not make use of it? This also addresses an apparent tension between our findings and previous work demonstrating that pretrained models contain rich linguistic information (Hewitt and Manning, 2019; Tenney et al., 2019b), which should be sufficient to at least solve Syntax-COGS.

5.1 Probing for structural information

We use the well-established probe task methodology (Peters et al., 2018; Tenney et al., 2019a) to analyze what information is present in the outputs of the BART encoder. We define both a syntactic and a semantic probing task:

Constituent labeling. The goal of this task is to predict correct labels for all constituency spans in a sentence. We treat spans that are not constituents

as if they were annotated with the *None* label. The gold annotations are derived from Syntax-COGS.

Semantic role labeling. To measure the presence of structural semantic information, we define a probe task that predicts role labels for all predicate-argument relations in a sentence. For example, in the sentence *Emma slept*, the goal is to recognize that *slept* is a predicate with *Emma* being its *agent*. This task captures most of the information in the original COGS meaning representations as relations between tokens in the sentence. We extract data for this task (given two tokens, predict if the second is an argument of the first and with what role label) from the COGS meaning representation. We refer to Appendix C for details.

We train probe classifiers in a similar way as (Tenney et al., 2019a). For each task, we train a multi-layer perceptron to predict the target label from the outputs of the frozen pretrained encoder. For constituent labeling, the MLP reads a span representation obtained by subtracting the encodings of the tokens at the span boundary from each other (Stern et al., 2017). For semantic role labeling, the input of the MLP is the concatenation of the encodings for the predicate and argument token.

We evaluate the probes in two ways. First, we train the probes on the original training split of COGS ("orig"). However, this conflates the presence of structural information in the encodings with the ability of the probing MLP itself to perform structural generalization. We therefore also evaluate on a second split ("probe") in which we add 60% of the generalization set (randomly selected) to the training set and 10% to the development set and keep the rest as the probe test set. This makes the probe test set in-distribution with respect to the probe training set. The encoder remains frozen and can therefore not adapt to the modified training set; we still obtain meaningful results about whether the pretrained encodings contain the information that is needed to learn to predict structure in COGS.

5.2 Results

We report the sentence-level accuracy in Table 2. For better comparison, all accuracies are measured on the test set from the "probe" split. We find that the probes learn to solve both tasks accurately on the "probe" split, indicating that the pretrained encodings of BART contain all the information that is needed to make structural predictions. By contrast, when we replace the BART encodings with random vectors of the same size ("Random" rows), the probe fails to learn. The probes also perform badly on the "orig" split, suggesting that the probe "decoder" does not generalize structurally either.

	Encoder	Data	STRUCT Obj to Subj PP	STRUCT CP recursion	STRUCT PP recursion	LEX 18 other types
sem	BART	probe	82	91	92	100
	Random	probe	25	0	65	90
	BART	orig	0	5	27	94
syn	BART	probe	85	80	83	100
	Random	probe	1	0	0	16
	BART	orig	0	0	7	92

Table 2: Exact match accuracy for probing on the individual generalization types.

These findings suggest that the BART encoder captures all the necessary information about the input sentence, but the BART decoder cannot use it to learn to generalize structurally.

5.3 Enriching seq2seq with structure

Can we make things easier for the decoder by making the structural information explicit in the input? To investigate this, we inject the gold syntax tree into the BART encoder to see if this improves structural generalization in semantic parsing.

We retrain BART on COGS, but instead of feeding it the raw sentence, we provide as input the linearized gold constituency tree ("`(NP (Det a) (N rose))`"), both for training and inference. This method is similar to Li et al. (2017) and Currey and Heafield (2019), but we allow attention over special tokens such as "`(`" during decoding.

We report the results as "BART+syn" in Table 1 and Fig. 3; the overall accuracy increases by 1.5% over BART. This is mostly because providing the syntax tree allows BART to generalize correctly on LEX. However, STRUCT remains out of reach for BART+syn, confirming the deep difficulty of structural generalization for seq2seq models.

We also explored other ways to inform BART with syntax, through multi-task learning (Sennrich et al., 2016; Currey and Heafield, 2019) and syntax-based masking in the self-attention encoder (Kim et al., 2021). Neither method substantially improved the accuracy of BART on the COGS generalization set (+1.0% and -6.4% overall accuracy, respectively). We conclude that the weakness of the BART decoder towards structural generalization persists even when the input makes the structure explicit.

	cp past	cp present
cc past	Oliver said **that Noah discovers a boy** and slept	Oliver says **that Noah discovered a boy and slept**
cc present	Oliver said **that Noah discovers a boy and sleeps**	Oliver says **that Noah discovered a boy** and sleeps

Question: What is the *ccomp* of *said/says*?

	pp singular	pp plural
rc singular	Noah ate the cakes beside **a plate** that was cooked	Noah ate **the cake beside plates** that was cooked
rc plural	Noah ate **the cakes beside a plate** that were cooked	Noah ate the cake beside **plates** that were cooked

Question: What is the *theme* of *cooked*?

Figure 5: Construction of QA-COGS-disamb: top is *cc_cp*, bottom is *rc_pp*. The answer to the example question is highlighted in bold.

6 Text-to-text structural generalization

We will now turn our attention to a novel text-to-text variant of COGS. The difficulty of structural generalization for seq2seq models has been primarily studied on tasks where sentences must be mapped into symbolic representations of some kind, such as the semantic and syntactic representations in Section 4. But although pretrained seq2seq models like BART and T5 achieve excellent accuracy on broad-coverage semantic parsing tasks, one might argue that they were originally designed for tasks where the output sequence is natural language as well, and thus should be evaluated on such tasks.

We therefore propose a new dataset, QA-COGS, which presents structural generalization examples based on COGS sentences in a question-answering format. Given a context sentence and a question sentence as input, the goal is to output the correct answer, which should be a consecutive span of tokens in the context sentence. The dataset consists of two sections: *QA-COGS-base* directly asks questions about COGS sentences (Section 6.1), whereas *QA-COGS-disamb* combines COGS sentences in novel coordinating structures (Section 6.2). Following the original COGS design, each section consists of four subsets: training set, development set, in-distribution test set, and out-of-distribution generalization set.

6.1 QA-COGS-base

The QA-COGS-base dataset uses the sentences of COGS as context sentences, and then asks one or more questions about each sentence that can be answered by a contiguous substring (see Fig. 6). For example, given *Noah ate the cake on the plate* as context, we ask *What did Noah eat?* and *Who ate the cake on the plate?*, and the answer should be *the cake on the plate* and *Noah* respectively.

To generate question-answer pairs, we identify the semantic roles and arguments for each predicate in all sentences of COGS, as in the SRL probing task (Section 5.1). We generate question-answer pairs out of these based on handwritten templates (i.e. at least one per COGS instance) and split them into train/test/generalization sets as in the original COGS. We refer to Appendix D for more details.

The original COGS training set contains "primitive" instances in which the sentence consists of a single word, and the meaning representation is the word itself (e.g. Paula ⇒ Paula). We include these instances in QA-COGS-base by using a special token <*prim*> as the question sentence and the primitive word as context and answer (i.e., Paula <*prim*> ⇒ Paula).

6.2 QA-COGS-disamb

We add QA-COGS-disamb as a second, harder text-to-text task based on COGS. This task exploits the interplay of the syntactic structure of a sentence with constraints on tense and number agreement. For instance, in sentences of the form "N1 V1 that N2 V2 and V3" (where N1, N2 are noun phrases and V1, V2, V3 are verbs), V3 belongs to the same clause as V1 or V2 depending on which one it agrees with. Thus, the agreement between verbs disambiguates a structural ambiguity of the sentence. Some concrete examples are shown in Fig. 5. The idea that agreement interacts with syntax is reminiscent of Linzen et al. (2016), but here we predict the syntactic structure rather than the agreement feature.

QA-COGS-disamb consists of two parts. The subcorpus *cc_cp* consists of sentences as above, where tense agreement disambiguates the structural ambiguity between CP embedding and coordination. The subcorpus *rc_pp* contains sentences where number agreement disambiguates the attachment of a relative clause. In both cases, we construct context sentences using a context-free grammar adapted from the one that generates COGS.

Gen type	Training	Generalization
Obj to Subj PP	Noah ate **the cake on the plate**. What did Noah eat?	**The cake on the plate** burned. What was burned?
PP recursion	Ava saw **a ball in a bowl on the table**. What did Ava see?	Ava saw **a ball in a bowl on the table on the floor**. What did Ava see?
CP recursion	Ava said **that Emma liked that a dog ran**. What did Ava say?	Ava said **that Emma liked that Noah noticed that a dog ran**. What did Ava say?

Figure 6: Examples for the QA-COGS-base dataset with regard to each structural generalization type. In each example, the first sentence is the context sentence, the second sentence is the question sentence and the bold token span is the corresponding answer.

		QA-COGS-base					QA-COGS-disamb	
		STRUCT			LEX			
Model Class	Model	Obj to Subj PP	CP recursion	PP recursion	all 18 other types	Overall	*cc_cp*	*rc_pp*
seq2seq	BART	99	59	69	95	86	37	14
	T5	100	95	97	100	99	16	22
structure-aware	BART-QA	100	98	100	100	99	6	0
	BART-QA+struct	-	-	-	–	-	100	100

Table 3: Exact match accuracy on the individual generalization types on the sections of QA-COGS.

We generate questions of the form "What is the ccomp of said?" along with their answers from the context sentences using a small number of hand-written heuristics. Answering these questions correctly amounts to disambiguating the structure of the sentence.

We create training (4k instances), development (1k), and in-domain test sets (1k) for QA-COGS-disamb out of three of the four combinations of the agreement features of the two verbs (white cells in Fig. 5). We create a generalization set (2k instances) from the fourth, unseen combination of agreement features (gray cells).

7 Experiments on QA-COGS

7.1 Models

We conduct a series of experiments in which a model receives the concatenation of context sentence and question as input and must predict the answer. We fine-tune BART and T5 on QA-COGS and compare against two structure-aware models. Details of the training setup are discussed in Appendix A.

First, we compare against an extractive model we call BART-QA. Given a context sentence and question, BART-QA predicts the start and end position of the answer within the context sentence. The start and end positions are each predicted by an MLP trained from scratch which takes the outputs of the pretrained BART encoder as input.

Second, we use a more informed model called BART-QA+struct specifically for QA-COGS-disamb. BART-QA+struct shares the same encoder as BART-QA, but its decoder is constrained to select a span which exists in the gold syntax tree of the sentence. This model accesses information that is usually not available at test time, and we offer it only as a point of comparison.

7.2 Results

The exact match accuracies on the generalization sets are shown in Table 3. Similar to the earlier experiments, all models perform well on LEX; we mainly discuss results on STRUCT below.

QA-COGS-base. All models solve "Object to Subject PP" perfectly, with T5 and BART-QA also achieving perfect accuracy on the PP and CP recursion. While these positive results on structural generalization seem to go against the grain of our earlier discussion, it is important to note that QA-COGS-base is an extractive task which only requires selecting a substring of the input; and further, that this substring is in a very specific position of the string, making the task amenable to learning simple heuristics (e.g. subject is everything to the left of the verb). Thus, these results indicate that structural generalization is hard only if the decoder's task is sufficiently complex. Note that unlike BART, T5 sees question answering tasks during training, which may help explain the difference in accuracy.

QA-COGS-disamb. However, BART and T5 all achieve low accuracy on QA-COGS-disamb, suggesting that even text-to-text tasks involving

structural generalization can be difficult; string-level heuristics are not successful on this task. In this case, the task is still hard for the structure-aware model BART-QA. It can be solved by BART-QA+struct, but note that this model has access to gold syntax information which makes the task much easier. Note that since the training and generalization sentences in QA-COGS-disamb are of similar length, the difficulty comes exclusively from structural rather than length generalization.

8 Conclusion

We have presented evidence that structural generalization is hard for seq2seq models, both on semantic and syntactic parsing (COGS and Syntax-COGS) and on some text-to-text tasks (QA-COGS-disamb). In many of these cases, structure-aware models generalize successfully where seq2seq models struggle. Unlike earlier work, we have shown that this effect persists when the seq2seq models can be pretrained.

We have then presented a number of experiments to help pinpoint the cause of this limitation. We found that the BART encoder still provides structural information, but the decoder does not use it to generalize – both in the parsing tasks and in the probing tasks on the original splits, and not even when the input is enriched with syntactic information. We further found that when the decoder's task is simple enough, as in QA-COGS-base, seq2seq models learn to generalize structurally as well as structure-aware models. In improving the ability of seq2seq models to generalize structurally, it seems promising to focus on the decoder, especially by including structure-aware elements.

9 Limitations

Our experiments are limited to a synthetic corpus (COGS) and its derivatives. While it seems plausible to us to justify negative results like ours with a synthetic corpus, it must be recognized that the distribution of language in COGS is not the same as in English as a whole, which might undermine the ability of both seq2seq and structure-aware models to learn to generalize.

Furthermore, claims about a whole class of models (seq2seq) can only be supported, never completely proved, through empirical experiments on a finite set of representatives. Nonetheless, we think that this paper has considered a sufficiently wide range of models and tasks to make careful statements about seq2seq models as a class.

Acknowledgements

We are indebted to Lucia Donatelli and Pia Weißenhorn for fruitful discussions, and to Najoung Kim for providing the code for generating COGS. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) through the project KO 2916/2-2.

References

Ekin Akyürek and Jacob Andreas. 2021. Lexicon learning for few shot sequence modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4934–4946, Online. Association for Computational Linguistics.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-21)*, volume 35, pages 12564–12573. AAAI Press.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. *arXiv preprint arXiv:2201.05899*.

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

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3322–3335, Online. Association for Computational Linguistics.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 619–634, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anna Currey and Kenneth Heafield. 2019. Incorporating source syntax into transformer-based neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 24–33, Florence, Italy. Association for Computational Linguistics.

Jerry A. Fodor and Ernest Lepore. 2002. *The Compositionality Papers*. Oxford University Press.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3–71.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *Computing Research Repository (CoRR)*, arXiv:2007.08970.

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

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*.

Juyong Kim, Pradeep Ravikumar, Joshua Ainslie, and Santiago Ontañón. 2021. Improving compositional generalization in classification tasks via structure annotations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 637–645, Online. Association for Computational Linguistics.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *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.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882, Stockholmsmässan, Stockholm Sweden. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697, Vancouver, Canada. Association for Computational Linguistics.

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

Chenyao Liu, Shengnan An, Zeqi Lin, Qian Liu, Bei Chen, Jian-Guang Lou, Lijie Wen, Nanning Zheng, and Dongmei Zhang. 2021. Learning algebraic recombination for compositional generalization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1129–1144, Online. Association for Computational Linguistics.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2020. Does syntax need to grow on trees? sources of hierarchical inductive bias in sequence-to-sequence networks. *Trans. Assoc. Comput. Linguistics*, 8:125–140.

Santiago Ontanon, Joshua Ainslie, Zachary Fisher, and Vaclav Cvicek. 2022. Making transformers solve compositional tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3591–3607, Dublin, Ireland. Association for Computational Linguistics.

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

Linlu Qiu, Peter Shaw, Panupong Pasupat, Paweł Krzysztof Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2021. Improving compositional generalization with latent structure and data augmentation. *arXiv preprint arXiv:2112.07610*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Pia Weißenhorn, Lucia Donatelli, and Alexander Koller. 2022. Compositional generalization with a broad-coverage semantic parser. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics (*SEM)*.

Xiang Yu, Ngoc Thang Vu, and Jonas Kuhn. 2019. Learning the Dyck language with attention-based Seq2Seq models. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 138–146, Florence, Italy. Association for Computational Linguistics.

Hao Zheng and Mirella Lapata. 2022. Disentangled sequence to sequence learning for compositional generalization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4256–4268, Dublin, Ireland. Association for Computational Linguistics.

A Training details

Evaluation metrics. We use sequence-level exact match accuracy as our evaluation metrics for all experiments. Thus a predicted sequence is correct only if each output token in it is correctly predicted.

Hyperparameters. We used the following hyperparameter values in our experiments. For all experiments we reported, we use *bart-base*[3] for BART model and *t5-base*[4] for T5 model. We always use the Adam optimizer (Kingma and Ba, 2015) and gradient accumulation steps 8. Exact match accuracy is used as the validation metric.

We use the same hyperparameters setting for semantic parsing, syntactic parsing and POS tagging experiments. For BART, we use batch size 64 and learning rate 2e-4. For T5, we use batch size 32 and learning rate 5e-4.

In probing experiments, we probe the encoder of BART. The hidden size of the MLP classifier is 1024 and the dropout is 0.1. We use batch size 64 and learning rate 1e-3 for the span prediction task and 1e-4 for the semantic role labeling task.

In the QA-COGS experiments, we adapt the question answering module[5] of BART for BART-QA and BART-QA+struct. In the evaluation for such extractive models, we do not consider the capitalization of the determiners (e.g. *The boy* is equivalent to *the boy*). We use batch size 64 and learning rate 2e-4 for these two models. For seq2seq models, most hyperparameters are the same as the ones in parsing tasks. The only difference is that we use learning rate 1e-4 for the T5 model.

Model selection. Csordás et al. (2021) find that using an in-distribution development set can lead to inefficient model selection and they select their best model based on the accuracy on the generalization set. We follow Zheng and Lapata (2022) by sampling a subset of the generalization set as an out-of-distribution development set.

In-distribution set performance. The exact match accuracy is at least 99 for both the (in-distribution) development set and the (in-distribution) test set in all experiments for the parsing and tagging tasks.

On the QA-COGS-base dataset, all models (i.e. BART, T5 and BART-QA) achieve at least 99 ac-

[3] https://huggingface.co/facebook/bart-base

[4] https://huggingface.co/t5-base

[5] https://huggingface.co/transformers/v4.5.1/model_doc/bart.html#transformers.BartForQuestionAnswering

curacy on the in-distribution development and test sets. On the QA-COGS-disamb dataset, we find T5 and BART-QA can achieve an accuracy of 100 on the in-distribution development and test sets across different randoms seeds. However, the performance of BART is not stable with regard to different random seeds. The mean accuracy averaged over 5 runs is 95 ± 8.1 for *cc_cp* and 73.8 ± 27.8 for *rc_pp*.

Other details. Training takes 4 hours for BART with about 50 epochs and 4 hours for T5 with about 30 epochs. Inference on the generalization set takes about 1 hour. All experiments are run on Tesla V100 GPU cards (32GB). The number of parameters is 140 million in BART and 220 million in T5.

Results from other papers. (Kim and Linzen, 2020) provides two train sets: *train* (24155 samples) and *train100* (39500 samples). The *train100* simply extends *train* with 100 samples for each exposure example. For example, for the generalization type in Fig. 1 (a), *train* set only contains 1 sentence with *hedgehog* being the subject as the exposure example, but *train100* contain 100 different sentences with *hedgehog* being the subject. Since *train100* does not introduce new structures, it is only used to help lexical generalization.

All semantic models in Table 1 are trained on the *train* set, except for (Kim and Linzen, 2020; Conklin et al., 2021; Weißenhorn et al., 2022). We noticed that (Kim and Linzen, 2020; Conklin et al., 2021) get higher performance on *train100* and thus report their number on *train100*. Although the number for (Weißenhorn et al., 2022) is based on *train100*, their model actually performs well on structural generalization when trained on the *train* set and using *train100* only improves the performance on lexical generalization types. Thus their model still supports the point that structural generalization can be solved by structure-aware models.

B Dataset details

We use COGS (Kim and Linzen, 2020) and variants of COGS (i.e. Syntax-COGS, POS-COGS and QA-COGS) as our datasets. We report dataset statistics for all our datasets in Table 4.

Syntactic annotations. To obtain syntactic annotations for Syntax-COGS, we use NLTK[6] to parse each sentence in COGS with the context-free grammar that was used to generate COGS. In our experiments, we find this parsing process yields a unique tree for each sentence in COGS. The original grammar contains rules such as `NP→NP_animate_dobj_noPP`. We replace such fine-grained nonterminals (e.g. `NP_animate_dobj_noPP`) with general nonterminals (e.g. `NP`). This results in duplicate patterns (e.g. `NP→NP`) and we further remove such patterns from the output tree.

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

C Semantic role labeling

We give more details about semantic role labeling task described in Section 5.1 here. In contrast to the semantic parsing task, where the output is a sequence encoding the meaning representation, the goal of this task is to predict the semantic role graph of a sentence.

An example of the semantic role graph is shown in Fig. 7. The symbol – denotes that the column word is not an argument of the row word; we capture this with the special class *None* in the data.

We align tokens in the sentence and predicate symbols in the meaning representation based on the variable names, which specify positions in the sentence (e.g. x_1 corresponds to the second token in the string). This allows us to project the predicate-argument relations in the meaning representation to relations between the tokens. For a predicate verb, we connect an edge to each of its arguments (i.e. *drew* has an `Agent` edge to *girl.*) in the meaning representation.

The COGS grammar also contains prepositional phrases (e.g. *a bowl on the table*). To represent this modification relation, we connect an `Nmod` edge from the modified noun to the modifier noun (e.g. *bowl* has an `Nmod` edge to *table*).

For common nouns, we connect a `DefN` edge to itself to denote it has a definite determiner (e.g. *girl* has a `DefN` edge to itself) and a `IndefN` to denote it has an indefinite determiner (e.g. *bat* has an `IndefN` edge to itself).

D QA-COGS

D.1 QA-COGS-base

To create QA-COGS-base, we first obtain the frame for each predicate in a sentence from its gold meaning representation. We define the frame of a predicate as the combination of argument types it takes. Possible frames in our dataset and corresponding examples are shown in Table 5. We generate questions for all predicate-argument pairs in

Dataset	# train	# dev.	# test	# gen	Vocab. size	Train len.	Gen len.
COGS	24155	3000	3000	21000	871	22/153	61/480
Syntax-COGS	24155	3000	3000	21000	759	22/129	61/375
POS-COGS	24155	3000	3000	21000	753	22/21	61/60
QA-COGS-base	54349	6834	6798	67989	793	44/19	123/57
cc_cp (4 splits)	4000	1000	1000	2000	709	36/25	35/18
	4000	1000	1000	2000	709	35/21	36/25
	4000	1000	1000	2000	709	36/25	30/19
	4000	1000	1000	2000	709	36/25	34/21
rc_pp (4 splits)	4000	1000	1000	2000	594	19/5	19/5
	4000	1000	1000	2000	594	19/5	19/2
	4000	1000	1000	2000	594	19/5	19/2
	4000	1000	1000	2000	594	19/5	19/5

Table 4: Statistics for all our datasets. # denotes the number of instances in the dataset. Vocab.size denotes the size of vocabulary for the dataset, which consists of input tokens and output tokens. Train.len denotes the maximum length of the input tokens and output tokens in the train set. Gen.len denotes the maximum length in the generalization set.

Input: The girl drew a bat .

MR: `*girl(x_1) ; draw.agent(x_2, x_1)`
`AND draw.theme(x_2, x_4) AND bat(x_4)`

Semantic role graph:

	The	girl	drew	a	bat
The	-	-	-	-	-
girl	-	DefN	-	-	-
drew	-	Agent	-	-	Theme
a	-	-	-	-	-
bat	-	-	-	-	IndefN

Figure 7: An example to show how to transform a meaning representation to a semantic role graph.

a sentence. Thus a sentence with two predicates both of which takes two arguments will result in 4 question-answer pairs.

D.2 QA-COGS-disamb

We adapt the original context-free grammar from which the COGS training set was generated and make some changes to it to generate QA-COGS-disamb. We refer readers to Appendix A and B in Kim and Linzen (2020) for more details of the original grammar.

For cc_cp, we introduce the coordination structure and present tense into the grammar. We also simplify the grammar by removing the grammar rule for passive verbs (e.g. *eaten*) and subject control verbs that take infinitival arguments (e.g. *try*). We do this to avoid such verbs resulting in ambiguous sentences (e.g. *Oliver said that Noah is helped and painted*). The grammar enforces that the tense of the complement clause must be different from the one in the main clause to avoid ambiguity (that is, if the verb in the main clause is in past tense, then the verb in the subordinate clause must be in present tense). We extend the verb vocabulary with their present tenses and use the same noun vocabulary as COGS.

For rc_pp, we add relative clauses and plural nouns to the grammar. We also simplify the grammar by removing the grammar rule for verb phrases that do not have a common noun as object, e.g. verbs taking CP arguments (e.g. *say*) and unac-

Predicate Frame	Context	Question	Answer
AGENT	The captain ate	Who ate ?	the captain
THEME	The donut was known	What was known ?	the donut
AGENT_THEME	Emma ate the ring beside a bed	What did Emma eat. ? Who ate the ring beside a bed ?	the ring beside a bed Emma
AGENT_THEME_RECIPIENT	Amelia gave Emma a strawberry	Who gave a strawberry to Emma ? What did Amelia give to Emma ? Who did Amelia give a strawberry to ?	Amelia a strawberry Emma
THEME_RECIPIENT	A rose was mailed to Isabella	Who was a rose mailed to ? What was mailed to Isabella ?	Isabella a rose
AGENT_CCOMP	Liam meant that Sophia rolled a teacher on a seat	What did Liam mean ? Who meant that Sophia rolled a teacher on a seat ?	that Sophia rolled a teacher on a seat Liam
AGENT_XCOMP	Emma hoped to run	Who hoped to run ? What did Emma hope to do ?	Emma run

Table 5: All possible predicate frames and corresponding question-answer examples for the QA-COGS-base dataset.

cusative verbs (e.g. *sleep*). The grammar enforces that the head noun of the NP and the head noun of the PP differ in number (that is, if the NP is singular, then the PP must be plural). In original COGS grammar, the vocabularies for nouns in NPs and PPs are separate (e.g. *a cake on the table*, *table* can only appear after *on*). We change this by using the same noun vocabulary for both. We also extend the noun vocabulary with their plural forms and extend the verb vocabulary with *were*.

E Detailed results

We report detailed results for our best models in Table 6. We report averaged accuracy and the standard deviation over 5 runs. *BART+mtl* and *BART+mask* denotes the model we used in section 5.3.

Dataset	Model	STRUCT Obj to Subj PP	STRUCT CP recursion	STRUCT PP recursion	LEX all 18 other types	overall
COGS	BART	0.0 ± 0.0	0.5 ± 0.2	11.8 ± 1.5	91.1 ± 0.4	78.6 ± 0.3
	BART+syn	0.0 ± 0.0	5.3 ± 0.8	7.7 ± 0.3	92.8 ± 0.5	80.1 ± 0.5
	BART+mtl	0.0 ± 0.0	0.3 ± 0.2	10.9 ± 1.9	92.1 ± 0.3	79.5 ± 0.3
	BART+mask	0.0 ± 0.0	0.1 ± 0.1	4.9 ± 2.9	84.0 ± 2.8	72.2 ± 2.5
	T5	0.0 ± 0.0	0.0 ± 0.0	8.6 ± 2.2	96.9 ± 0.3	83.5 ± 0.2
Syntax-COGS	BART	0.0 ± 0.0	9.1 ± 1.5	22.3 ± 1.1	99.5 ± 0.1	86.8 ± 0.1
	T5	4.7 ± 8.7	7.2 ± 1.3	9.0 ± 4.0	99.4 ± 0.5	86.2 ± 0.3
POS-COGS	BART	0.0 ± 0.0	5.8 ± 5.2	19.1 ± 10.2	97.9 ± 1.0	85.1 ± 1.1
	T5	0.0 ± 0.0	4.2 ± 2.5	3.9 ± 4.1	98.1 ± 1.1	84.5 ± 0.9
QA-COGS-base	BART	98.9 ± 0.7	58.8 ± 4.3	69.1 ± 1.0	95.3 ± 0.3	85.7 ± 0.7
	T5	100.0 ± 0.0	94.7 ± 2.3	96.9 ± 0.6	100.0 ± 0.0	98.6 ± 0.5
	BART-QA	100.0 ± 0.0	97.6 ± 0.9	99.6 ± 1.0	100.0 ± 0.0	99.2 ± 0.4
cc_cp	BART	-	-	-	-	36.5 ± 22.0
	T5	-	-	-	-	15.6 ± 1.7
	BART-QA	-	-	-	-	5.6 ± 10.0
	BART-QA+struct	-	-	-	-	100.0 ± 0.0
rc_pp	BART	-	-	-	-	13.7 ± 5.1
	T5	-	-	-	-	21.9 ± 2.0
	BART-QA	-	-	-	-	0.0 ± 0.0
	BART-QA+struct	-	-	-	-	100.0 ± 0.0

Table 6: Detailed results for our models across COGS, Syntax-COGS, POS-COGS and QA-COGS.