A case for pre-training in Compositional Generalization tasks

Stanford CS224N Custom Project

Ahmad Jabbar
Department of Linguistics
Stanford University
jabbar@stanford.edu

Rhea Kapur
Department of Linguistics
Stanford University
rheak@stanford.edu

Abstract

Do neural models have the capacity to compositionally generalize (Montague, 1970; Fodor and Pylyshyn, 1988; Janssen, 1997)? We run two experiments. With Experiment 1, we make a case for using pre-trained models for compositional generalization. We use results in Experiment 1 to motivate Experiment 2, where we finetuning two different versions of the pretrained T5 model (T5-small and T5-base) on ReCOGS (Wu et al., 2023). We report higher accuracies on some of the generalization tasks using T5-base than the baselines presented in (Wu et al., 2023). This is a promising result, and it sets us up to use constrained models (Timkey and Linzen, 2023) on ReCOGS in the future, with the motivation of exploring how a well-crafted dataset can lead to higher accuracy on the task even for an attention-constrained model.

1 Key Information to include

- Mentor: Shikhar Murty
- External Collaborators (if you have any): Christopher Potts
- Sharing project: No
- Team Contribution Statement: Chris and Ahmad came up with the idea of Experiment 2. Ahmad came up with the idea of Experiment 1. Rhea implemented and ran Experiment 2. Ahmad implemented and ran Experiment 1. Both Ahmad and Rhea contributed to the writing. Shikhar suggested using T5-base and provided feedback.

2 Introduction

Compositional generalization tasks such as COGS and SCAN (Lake and Baroni, 2018; Kim and Linzen, 2020; Wu et al., 2023; Lake and Baroni, 2023) seek to test the ability of language models to compositionally generalize. It's best to start with a thorough understanding of compositional generalization and compositionality more generally. For every novel sentence S that humans encounter, the meaning for S is not learned, but constructed on the fly by some prior knowledge of the lexical items and the structure in which those items are arranged in S. The capacity that makes such a feat possible may be understood as the capacity to compositionally generalize. A precise way to think about compositionality is as a mapping between the syntax of a language and its semantics. Even more precisely, following Montague (1970); Dowty (2007) take there to be two algebras.

(1) $\langle \alpha, F_\gamma \rangle_{\gamma \in \Gamma}$
α is a set of primitive and derived expressions that is closed under all operations F. Each subscripted F denotes a different syntactic operation.

(2) $\langle \beta, G_\gamma \rangle_{\gamma \in \Gamma}$
β is a set of primitive and derived meanings that is closed under all operations G. As for F, each subscripted G denotes a different semantic operation.

To illustrate, $F_1(\alpha, \beta)$ can be taken to arrange α and β in a precedence relation. Then, a system is compositional if and only if for F_1, and every such operation, there is a corresponding G_1 that has as its operands the meanings of the expressions that are F's operands. This homomorphism can more formally be captured as in (3).

(3) $h(F(\alpha_1, ...\alpha_n)) = G(h(\alpha_1), ...h(\alpha_n))$

Thinking of compositionality as a homomorphism is helpful also because it provides us with a clearer understanding of tasks like SCAN and COGS. While for instance, in SCAN, each expression of the input language is mapped to a meaning in the output language, COGS and ReCOGS only require the model to have learned the algorithm for semantic composition. This is most clearly seen in the variance between COGS and SCAN, in how primitives in the input language are mapped. For instance, while *jump* in SCAN is mapped to JUMP, *Lina* in COGS is mapped to *Lina*. This difference is brought about most clearly in the recent Lake and Baroni (2023), where the action sequences are removed by colored shapes. For now, we can look at (1), (2), and (3), and think of COGS as testing the model's ability to generalize and construct a $\langle \beta, G_\gamma \rangle_{\gamma \in \Gamma}$ from the train-set that can help it correctly map input sequences to their LFs at test time. Here, LFs can be thought of as primitive and complex expressions in $\langle \beta, G_\gamma \rangle_{\gamma \in \Gamma}$.

With the helpful analytical tools at hand and a conceptual understanding of compositionality, we can zoom in on how this capacity is operationalized in COGS and ReCOGS, the two benchmarks on which we'll focus for this paper. COGS and ReCOGS operationalize compositionality in terms of the capacity of the model to map an input sequence of a natural language string to an output sequence of a logical form, as done by Kim and Linzen (2020). (4)-(5) is an illustrative example for this task, where the model sees pairings like (4) and (5), and is asked to predict LFs for constructions not seen at train-time:

(4) Emma ate the cake on the table.

(5) `*cake(x_3); *table(x_6); eat.agent(x_1, Emma) AND eat.theme(x_1, x_3) AND cake.nmod.on(x_3, x_6)`

In terms of testing model performance, the above can be operationalized in terms of a seq2seq task. A model with an encoder and a decoder provides the most natural setting for such a task then. The encoder receives the input string as in (4) and the decoder aims to predict the gold LF. In the literature, the model accuracy is measured by averaging exact matches of the gold LFs across a range of test examples. There can be methodological concerns w.r.t. averaging over exact matches as being the best approach to measure model accuracy. For instance, it's quite possible that the model approximates well to all gold LFs, but overall performs poorly for exact matching. That this alternative approximating strategy is an optimal strategy is extremely likely under the assumption that language models are resource-rational agents (Lieder and Griffiths, 2020; Icard, 2023). Although we note these shortcomings for exact match based accuracy metrics, for this paper, we assume the metric, as the methodological concerns regarding accuracy metrics is orthogonal to our project.

In this paper, we're interested in the extent to which pre-training can be helpful in getting traction on some generalization tasks within COGS for which the literature has reported only very low accuracies. Zheng and Lapata (2021) use a pre-trained model on COGS and do report high accuracies. How is our contribution novel then? Our contribution is novel in two respects:

(6) We evaluate a pre-trained model's performance on ReCOGS rather than on COGS.

(7) We run experiments aimed at showing that a model trained from scratch on a task like COGS fails at compositional generalization, not because of its inability to build compositional representations for certain syntactic structures, but because it is not able to build the syntactic structures themselves in the first place.

(7) is stated at a very high level of abstraction. A lower-level description of (7) is available due to the use of surprisal (Shannon, 1948) as a proxy for grammaticality (Smith and Levy, 2013; Futrell et al., 2019, 2020). We aim to show that the model performs poorly to predict the gold LF for an input sequence because the model is surprised to see the input sequence. We lay out the details for both of these experiments extensively below. For now, in order to understand why (6) is significant, we give the reader a brief introduction to ReCOGS.

3 Related Work

Tasks for compositional generalization are usually touted as difficult for present-day models. Wu et al. (2023) show that these claims are inflated and the model failures are in part due to incidental features

of the benchmarks. More specifically, poor performance on COGS is due to incidental features of the LF representations as in (5) that can be removed while preserving semantic identity of the LF. Wu et al. (2023) run three experiments on two models: (i) LSTM (9M parameters) with a 2-layer LSTM encoder with global attention and a 2-layer LSTM decoder with a 512 hidden dimension size; (ii) Transformer (4M parameters) with a 2-layer Transformer blocks with 4 attention heads and a 300 hidden dimension size. Wu et al. (2023) modify COGS and run three experiments. Modified COGS comprises ReCOGS, a new benchmark. The contributions can be summarized as follows:

LFs in COGS like (5) contain redundant tokens like underscores and variable names. Removing these tokens preserves semantic identity of the LF. In addition, in COGS, the bigram (`, x`) appears approximately 10 times more than the next most frequent bigram (`, Emma`). The resulting model distributions are highly skewed, which can be assuaged by removing the most frequent token. As the overall model prediction improves, the prediction for the right LFs improves too. This falls out of the redundant token removal strategy employed.

One of the tasks in COGS requires the model to generalize to sequences of recursive depth not seen at train-time. This can involve CP-recursion such as *John knows that Mary thinks that Bill believes that Mina came to the party* or PP-recursion. Deeper CP-recursion correlates with longer sequence length. This gives rise to a confound. The model's inability to predict LFs for longer strings will interfere with its ability to predict gold LFs. To remedy this confound, examples already existing in the train-set are concatenated such that the corresponding LFs come out to have variable names that correspond to higher numerical values. Adding 1000 such concatenated examples improves the transformer performance significantly with lower returns on more examples. LSTM performance keeps improving until the highest train-set augmentation comprising 3072 examples.

Kim and Linzen (2020) report poor performance on *Obj PP* $\rightarrow$ *Subj PP*. This task requires the model to predict gold LFs for sentences with PP-modifiers, like *on the table*, modifying the noun in subject position, when in the train-set, the model only sees PP-modifiers modifying the nouns in subject position. In other words, the model sees natural language strings like (8) in the train-set, but is asked to predict LFs for strings like (9).

(8) Emma ate the cake on the table.

(9) The cake on the table burned.

Without letting the model see subject PPs, Wu et al. (2023) induce the biases required for the model to generalize to LFs for sentences with subject PPs. Three strategies are employed. To take one such strategy, for 5% of the training examples, the modified phrases are preposed as in (10) so that the model, while only seeing corresponding LFs for object PPs, sees a lower value for the variable names associated with the modified object nouns.

(10) The cake on the table, Emma ate.

Another strategy is to add participial verb phrases for both subjects and objects. The model now learns a new type of modifier and it sees subjects being modified with this modifier. Note crucially that the model still doesn't see the modifier associated with PP modifiers `nmod` taking the variable associated with the subject as its first argument. The model performance sees significant gains due to inclusion of these heuristics in the train-set.

We note that adding heuristics to induce biases that increase model accuracy provides the model with a portion of the same knowledge that pre-training provides it with. Therefore, we aim to see how much boost in model accuracy pre-training affords. Further, one of the aims of Wu et al. (2023) is to remove the incidental features of the LFs that deflate model accuracies. With these incidental features removed, we arrive at a benchmark that is more faithful to the cognitive task of compositional generalization. The contribution due to Experiment 2 can be summarized as: gauging gains due to pre-training on a fairer benchmark. But first, we make the case for pre-training stronger by running Experiment 1.

4 Approach

Experiment 1 (Approach) : For this experiment, our aim is to use a model that most closely resembles the architectures in Kim and Linzen (2020) and Wu et al. (2023). Moreover, given that COGS and ReCOGS present a seq2seq task and given that we want to understand how the model

processes the input sequence, we ideally want to assess the information to which the encoder has access. For this reason, we choose to use TinyBERT (Turc et al., 2019; Bhargava et al., 2021) imported from the HuggingFace library. Note that this experiment only makes a case for pre-training, and doesn't itself use a pre-trained model. Accordingly, we re-initialized the weights of TinyBERT to random parameters, to only afford access to its architecture as an encoder model that can be straightforwardly trained on an unsupervised masking task on a new dataset. TinyBERT has 4.4M parameters, which is close to the number of the parameters (4M) for the transformer architecture used in Wu et al. (2023).

Experiment 2 (Approach) : For this experiment, our aim is to gauge the traction a pre-trained model gets when fine-tuned on ReCOGS. We run this experiment using two pre-trained models. We choose T5-small and T5-base. We import T5-small (60 million parameters) and T5-base (220 million parameters) (Raffel et al., 2020) from the HuggingFace library, and fine-tune each to the ReCOGS dataset. The T5 architecture is apt for this task as it is an encoder-decoder model. Although the model size of T5-base is a far cry from TinyBERT used in Experiment 1 and the model sizes in Kim and Linzen (2020) and Wu et al. (2023), T5-base is smaller in the context of pre-trained models—for comparision, GPT2-medium comprises approximately 350M parameters. We take its size *in the context* of pre-trained models to justify our choice. The same considerations hold for T5-small *a fortiori*.

We are using the accuracy scores reported in Wu et al. (2023) as our baseline; specifically, we compare against their previously reported accuracies on the test set and on non-lexical generalization tasks. The finetuning code was written ourselves, but some of the data preprocessing, setup, and tokenization code was borrowed from the CS224U repository (exact file). Additionally, for evaluation, we used part of the script in ReCOGS/run$_c$ogs.py from the GitHub repository associated with Wu et al. (2023) as well as the recogs$_e$xact$_m$atch() function in the CS224U repository (exact file). We used the CS224U ReCOGS processing and evaluation code with the knowledge of Chris Potts and Zhengxuan Wu (both authors on the ReCOGS paper). Any additional code was implemented ourselves.

5 Experiments

5.1 Experiment 1

5.1.1 Data

For experiment 1, we construct our dataset from the input sequences like (4) of COGS (available najoungkim/COGS/data/train.tsv). Note crucially, that we exclude sentences like (5) from this dataset. Only retrieving the input sequences of COGS leaves us with a dataset of approximately 24,154 sentences. This is only appropriate, as the aim here is to train the re-initialized TinyBERT model on the unsupervised task of predicting the next token in the dataset. This enables the model to build a probability distribution over the input vocabulary of COGS, via which we can then retrieve surprisal scores.

5.1.2 Evaluation method

Our experimental framework gives rise to two aims. First, the aim is to then gauge if these surprisal scores for a particular generalization co-vary with the accuracy on that generalization task—a metric we can retrieve from Kim and Linzen (2020). Second, do the surprisal scores vary for sequences in the train set and the test set? For instance, (11) is a train-set example and (12) is a test-set example.

(11) Emma ate the cake on the table.

(12) The cake on the table burned.

We mask the target site for each generalization as in (13) and (14) and obtain the surprisal score for the target word, i.e., *on* for (13) and (14) for the mask. Moreover, as we're using a bi-directional encoder model, we provide the context after the [MASK] token as well to enable the model to make better predictions.

(13) Emma ate the cake [MASK] the table.

(14) The cake [MASK] the table burned

A note on target sites and generalizations: For this experiment, we focus primarily on structural generalization tasks, leaving out lexical ones. One of these generalizations is ObjPP $\rightarrow$ SubjPP as exemplified in (11) and (12). The other two are generalizing to higher recursion depth for CP recursion as in (15a)-(15b) and generalizing to higher recursion depth for PP recursion as in (16a)-(16b).

(15) a. John thinks that it is raining.
b. John thinks that Mary believes that Bill knows that it is raining.

(16) a. John saw the ball in the bottle.
b. John saw the ball in the bottle on the box in the living room.

For CP-recursion sentences, we took the last complementizer, i.e., *that*, to be the target site and masked it. We then retrieved the surprisal scores for the complementizer *that*. For PP-recursion sentences, we took the last preposition to be the target site and masked it. We retrieved the surprisal scores for the preposition *in*. To construct masked examples like (13) and (14), we used ChatGPT. The reason why we constructed our own examples is because we believe that the model could have overfit to the input sequences in COGS, which can be a confound, by deflating the surprisal scores on sentences like (13) that the model has seen. Using ChatGPT, we constructed 12 lists of examples.

With the surprisal scores in hand, we gauge if the model is more surprised by examples like (14) as compared to (13). If this prediction is borne out, then an explanation for why the model performs poorly in mapping gold LFs for sentences like (12) in comparison to (11) can be explained in terms of processing difficulty of the input sequences of the test-set in COGS. Then, a way to level the playing-field between sentences like (11) and (12) in terms of surprisal is to bring pre-training into the mix. Pre-training will expose the model to input sequences of the form (12), and note crucially that by using a pre-trained model, we still don't expose the model to the gold LFs associated with such input sequences. Therefore, the task of mapping to gold LFs still remains a legitimate task *pace* Kim et al. (2022).

5.1.3 Experimental details

We import TinyBERT from the HuggingFace library. This is a model with 4.4M parameters. We re-initialize its weights and train it from scratch on our dataset. The model is trained for a total of 3 epochs on 24,154 examples with a training-rate of 5×10^{-5}. The batch size is set to 128.

5.1.4 Results

Here, we report on each generalization separately. First, we see a difference in the average surprisal over 100 strings like (11) that insantiate the ObjPP structure and the average surprisal over 100 strings like (12) that instantiate the SubjPP structure.

Generalization: *ObjPP* $\rightarrow$ *SubjPP*	
Structure	**Surprisal**
ObjPP	5.46
SubjPP	8.95

Table 1: Surprisal scores for structure (*ObjPP*) the model gets trained on and the structure (*SubjPP*) the model is expected to generalize to.

The second structural generalization requires the model to predict LFs for CP structures with higher recursive depth. In COGS, at train-time, the model only sees CP structures with recursive depth 1 and 2. At test time, the model has to predict LFs for CP-recursive structures of depth 3-12. We report mean surprisals for recursive depths 1-4, to illustrate the point that with each recursive depth, the surprisal of the complementizer *that* at the target site increases.
The third structural generalization of interest is PP-recursion. The model only sees sequences with one or two PP-modifiers in a recursive structure as in (16a). The model is expected to generalize to deeper PP-recursive structures like (16b). We obtained mean surprisal scores for PP-recursive structures of depth 1-6.

Generalization: *CP* recursion	
CP recursive depth	**Surprisal**
1	3.74
2	6.86
3	8.39
4	9.51

Table 2: Surprisal scores increase with each recursive depth of the embedded CP.

Generalization: *PP* recursion	
PP recursive depth	**Surprisal**
1	4.75
2	4.13
3	4.66
4	5.46
5	8.56
6	10.64

Table 3: Surprisal scores increase with each recursive depth after 3 of the embedded PP.

The above results are interesting because the surprisal scores first decrease for each recursive depth until 3, but then we see the same pattern as we see with CPs: the surprisals increase with each recursive depth. As the model is not only expected to generalize to predicting LFs for PP-recursive sequences of depth 3, but a range of them from 3-12, this result also corroborates our hypothesis about how the surprisal for input sequences in COGS train-set and the surprisal for input sequences in the test-set will vary. **Upshot**: If we think of the homomorphism in (3), our results suggest that the bad model performance on COGS is not solely due to the inability of the model to compositionally generalize by constructing a homomorphism. Instead, taking surprisal to be a proxy for grammaticality, the model is unable to construct the requisite syntactic structure for the input sequences that it sees only at test-time. To fix this and to see if we can fix this, we introduce pre-training into the mix. Although pre-trained models have been used before as in Zheng and Lapata (2021), we have provided a justification for using pre-trained models via the results presented above.

5.2 Experiment 2

5.2.1 Data

For Experiment 2, we use ReCOGS (Wu et al., 2023) which, as described above, is a benchmark for compositional generalization. It contains a suite of distinct tasks all aimed at testing the model's capacity to generalize. See (4)-(5) from above for an illustrative example of its pairings. The training set of ReCOGS consists of 135,547 pairs of a natural language string and a gold LF (as in (9) and (10)). We used the training set for finetuning because it is set up in the way of helping models learn relative positional indices. The test set contains 3,000 pairs and was used for our evaluation. We also did evaluation on the generalization set, which consists of 21,000 pairs and is split into a number of tests on different generalization tasks (ex. *Obj PP* $\rightarrow$ *Subj PP*).

5.2.2 Evaluation method

For evaluating both the fine-tuned T5-small and T5-base, we use the same metric described in the ReCOGS paper (Wu et al., 2023) (and which their authors also used for evaluation, as noted in the previous section). It generates percent exact string identity of logical forms (model output vs. ground truth) along the generalization splits present in ReCOGS. This is done via the recogs$_e$xact$_m$atch() function mentioned in the above section on the approach for Experiment 2. We evaluate only structural tasks (ObjPP $\rightarrow$ SubjPP, CP recursion, PP recursion, Subj $\rightarrow$ Obj Proper, Prim $\rightarrow$ Obj Proper, Prim $\rightarrow$ Subj Proper) due to compute constraints (full evaluation of the generalization set would have taken upwards of twelve hours with an A100 GPU). We also generate an overall (averaged) test accuracy on ReCOGS.

5.2.3 Experimental details

After retrieving from HuggingFace, we fine-tuned T5-small (60 million parameters) and T5-base (220 million parameters) for 4 epochs on the training set of ReCOGS Raffel et al. (2020). This took about 1.5 hours and 4 hours respectively on a Colab A100 High-RAM GPU instance. We used a learning rate of 3e-4 and a batch size of 64. Additionally, we monitored the TensorFlow Dashboard to ensure that overfitting was not an issue (early stopping also ensured as such). We then ran the evaluation script described above to generate the results.

5.2.4 Results

Table 1 reports our results in the third and fourth columns. We also provide a comparison with the performances so far reported in the literature for COGS and ReCOGS. Wu et al. (2023) run an LSTM model in addition to a Transformer; we report accuracy scores only for the Transformer. This is a fairer comparison, given that we are also using T5-small, a pre-trained transformer. The results for Kim and Linzen (2020) have been taken directly from their paper, whereas scores in Wu et al. (2023) have been estimated from a figure in Wu et al. (2023). In the code for our milestone, there was an error in fine-tuning T5-small where the data mistakenly included the some of the generalization set, leading to higher accuracies than was correct. We have since fixed this error for the final paper, and we now see that T5-small does not learn patterns in ReCOGS adequately even after sufficient finetuning (lowest scores in generalization tasks and very low test accuracy score). This is an interesting result; it seems that larger pretrained models are needed for this task. The T5-base performance corroborates this; we see **comparable test accuracy** (>90%) to what was reported in Wu et al. (2023) and **substantially better performance** on some generalization tasks, namely *Obj PP* → *Subj PP* (39% improvement), *Subj* → *Obj Proper* (85% improvement), and *Prim* → *Obj Proper* (33% improvement). While we were not able to fine-tune T5-large due to constrained compute resources, we expect that T5-large will give even better performance than what we have seen so far, following this trend. We will test this by fine-tuning T5-large over the summer as we continue the project.

Results				
Generalization	COGS Kim and Linzen (2020)	ReCOGS Wu et al. (2023)	ReCOGS+T5-small	ReCOGS+T5-base
ObjPP → SubjPP	0.00	0.04	0.22	**0.61**
CP recursion	0.00	**0.10**	0.00	0.02
PP recursion	**0.09**	0.02	0.00	0.02
Subj → Obj Proper	0.54	0.70	0.19	**0.85**
Prim → Obj Proper	0.23	0.50	0.29	**0.83**
Prim → Subj Proper	0.24	**0.90**	0.21	0.83
Test	**0.97**	0.93	0.31	0.91

Table 4: Accuracy scores for structural generalization tasks and test set.

6 Analysis

The results from both experiments complement each other synergistically. First, Experiment 1 corroborates our hypothesis that surprisal for target sites in each structure depends on whether the model has been exposed to that structure or not. Take for instance, *Obj PP* structures. The surprisal at the target site for such structures is much lower than the surprisal at the target site for *Subj PP* structures. Both Kim and Linzen (2020) and Wu et al. (2023) report low accuracies for *Obj PP* → *Subj PP*. If our hypothesis is right, then using pre-trained models should see increased accuracy on this generalization. Indeed that is what we report by using T5-small and T5-base. We go from Kim and Linzen (2020)'s 0.00 and Wu et al. (2023)'s 0.04 to **0.61** for T5-base.

It is true however that even using T5-base, a pre-trained model, on ReCOGS doesn't get much traction on CP-recursion and PP-recursion. Nonetheless, the results from Experiment 1 report that with each recursive depth, the surprisal increases. Here, we offer an explanation. It's quite possible that the long CP and PP recursive structures don't exist out there in the wild, and more specifically in the corpus

that T5-base is trained on. For instance, it is hard to imagine that the corpus includes a sentence with CP-recursive structure of depth 10. This further corroborates our hypothesis from Experiment 1. As even pre-trained models aren't exposed to sequences of such recursive depth, using pre-trained models cannot reduce the surprisal at seeing such structures.

To blinker ourselves to Experiment 2 now, we glean from the results in Table 4 that pretrained T5 models struggle on CP and PP recursion generalization tasks most of all (mostly looking at T5-base here, and choosing not to focus on test accuracy because it seems likely that when we test T5-large, that will reach near 100%). By examining (wrong) outputs generated by T5-base for examples in the test set as well as the CP and PP recursion generalization splits, it is clear that T5-base when finetuned still has gaps in learning relative positional indices. We have included screenshots of such errors in the appendix (see 1), but the main thing to note is that a few very specific numbers (representing certain positional indices) are repeated among many pairings in the test set, CP recursion, and PP recursion set where the model failed; instead, the numbers or positional indices should be varied and highly specific to each example. In particular, this pattern is exacerbated with the CP and PP recursion generalization tasks (both of which do not have an accuracy exceeding 2% with T5-base).

In terms of next steps from here, we are curious to see whether CP and PP recusion accuracies will improve with T5-large. If not, more in-depth evaluation and perhaps restructuring of the finetuning data may be needed.

7 Conclusion

Through Experiments 1 and 2, we make a case for using pre-trained models for compositional generalization and then show that pre-trained models can in fact lead to better performance on difficult generalization tasks. We fine-tune two different versions of the pretrained T5 model (T5-small and T5-base) on ReCOGS (Wu et al., 2023). We report higher accuracies on certain generalization tasks using T5 than with the baselines presented in (Wu et al., 2023), and comparable accuracy on the test set. This is a novel result, and it sets us up to use constrained models (Timkey and Linzen, 2023) on ReCOGS in the future, with the motivation of exploring how a well-crafted dataset can lead to higher accuracy on the task even for an attention-constrained model. We will also explore fine-tuning even larger pretrained models such as T5-large in the future and expect to see even better results. Moreover, we think that lower accuracies on CP recursion and PP recursion further corroborate our hypothesis: that the model ought to be exposed to input sequences of the test examples for the seq2seq task of compositional generalization. We think that using sequences like CP-recursive structures of depth 12 that are unlikely to occur on the web in most corpora is unfair. Note crucially that via pre-training, we don't expose the model to the gold LFs; thus we preserve the integrity of the generalization task. Our experiments give rise to new ideas for a fairer benchmark for compositional generalization. We seek to pursue construction of such a benchmark in the future.

8 Ethics Statement

We wish to comment on two aspects of our experiment. First, we think that we're resolving a theoretical question while using some compute that can perhaps be better used for experiments that are more practical and fruitful. However, we think that this is a concern that can be raised for any research question. For any research question Q, there's perhaps another question P that serves humanity more. This connects to objections against Utilitarianism more generally (Wolf, 1982; Feldman, 1986). Nonetheless, the question of using computing resources is germane. Via running this experiment, we are contributing to our carbon footprint. However, these concerns are also assuaged, given that most of the models that we are using for our experiments are pretrained. So, we don't have to use the compute required for pretraining. Another crucial contribution of our project can be noted via the following observation in Lake and Baroni (2018): that perhaps the inability of present-day models to systematically generalize may be responsible for their data-thirst. If it turns out that we can contribute in some way to understanding systematic generalization better, we can save up on future compute and money spent on extra training on extra data.

We noted above that our work has a theoretical bent. Nonetheless, our research has implications for understanding how humans communicate through novel complex sentence forms. As we stated earlier in this proposal, there are various ambiguities in understanding complex sentence forms (in

"Emma ate the cake on the table," is Emma on the table or was the cake on the table?). We will ensure that the various ambiguities are well-represented in our training data and that there is not any bias toward one particular interpretation (or that the bias is proportional to how often the interpretations appear in English or among English speakers).

AI use statement: we used ChatGPT for problem-solving purposes and mainly to generate sentences used for Experiment 1. This was done by providing it with an example and asking it to replicate it with certain variables. This was done to have a large sample size for each type of sentence. We did NOT use any AI tools to aid us in the writing of this report.

References

Bhargava, P., Drozd, A., and Rogers, A. (2021). Generalization in nli: Ways (not) to go beyond simple heuristics.

Dowty, D. (2007). Compositionality as an empirical problem. *Direct compositionality*, 14:23–101.

Feldman, F. (1986). *Doing the best we can: An essay in informal deontic logic*, volume 35. Springer Science & Business Media.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Futrell, R., Gibson, E., and Levy, R. P. (2020). Lossy-context surprisal: An information-theoretic model of memory effects in sentence processing. *Cognitive science*, 44(3):e12814.

Futrell, R., Wilcox, E., Morita, T., Qian, P., Ballesteros, M., and Levy, R. (2019). Neural language models as psycholinguistic subjects: Representations of syntactic state. *arXiv preprint arXiv:1903.03260*.

Icard, T. F. (2023). *Resource rationality*. Manuscript, Stanford.

Janssen, T. (1997). Compositionality. In Janssen, T. M. and Partee, B. H., editors, *Handbook of logic and language*, pages 417–473. Elsevier.

Kim, N. and Linzen, T. (2020). Cogs: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105.

Kim, N., Linzen, T., and Smolensky, P. (2022). Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models. *arXiv preprint arXiv:2212.10769*.

Lake, B. and Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Lake, B. M. and Baroni, M. (2023). Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121.

Lieder, F. and Griffiths, T. L. (2020). Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and brain sciences*, 43:e1.

Montague, R. (1970). Universal grammar. *Theoria*, 36:373–98.

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

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

Smith, N. J. and Levy, R. (2013). The effect of word predictability on reading time is logarithmic. *Cognition*, 128(3):302–319.

Timkey, W. and Linzen, T. (2023). A language model with limited memory capacity captures interference in human sentence processing. *arXiv preprint arXiv:2310.16142*.

Turc, I., Chang, M., Lee, K., and Toutanova, K. (2019). Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Wolf, S. (1982). Moral saints. *The Journal of Philosophy*, 79(8):419–439.

Wu, Z., Manning, C. D., and Potts, C. (2023). Recogs: How incidental details of a logical form overshadow an evaluation of semantic interpretation. *Transactions of the Association for Computational Linguistics*, 11:1719–1733.

Zheng, H. and Lapata, M. (2021). Disentangled sequence to sequence learning for compositional generalization. *arXiv preprint arXiv:2110.04655*.

A Appendix

Input

```
0          Mila liked that the cake was offered to Emma .
23                       Emma sent William the cookie .
30             Emma tolerated a champion beside a table .
45       Emma was lended a strawberry beside the book b...
54       A bear liked that Ella was sold the pencil in ...
                              ...
2954     Emma liked that the cake was returned to the c...
2970       The bunny liked that a rose was drawn by Emma .
2973               Zoe lended Emma the balloon in a car .
2988     A girl liked that the pillow was given to Broo...
2997            Noah adored the princess beside a bucket .
```

Output

```
0       Mila ( 37 ) ; * cake ( 52 ) ; Emma ( 10 ) ; li...
23      Emma ( 2 ) ; William ( 0 ) ; * cookie ( 51 ) ;...
30      Emma ( 2 ) ; champion ( 13 ) ; table ( 4 ) ; n...
45      Emma ( 11 ) ; strawberry ( 43 ) ; * book ( 14 ...
54      bear ( 40 ) ; Ella ( 49 ) ; * pencil ( 51 ) ; ...
                              ...
2954    Emma ( 8 ) ; * cake ( 4 ) ; * creature ( 21 ) ...
2970    * bunny ( 4 ) ; rose ( 20 ) ; Emma ( 49 ) ; li...
2973    Zoe ( 38 ) ; Emma ( 54 ) ; * balloon ( 52 ) ; ...
2988    girl ( 4 ) ; * pillow ( 22 ) ; Brooklyn ( 14 )...
2997    Noah ( 38 ) ; * princess ( 24 ) ; bucket ( 32 ...
```

Prediction

```
0       Mila ( 59 ) ; * cake ( 0 ) ; Emma ( 44 ) ; lik...
23      Emma ( 59 ) ; William ( 44 ) ; * cookie ( 0 ) ...
30      Emma ( 59 ) ; champion ( 44 ) ; table ( 0 ) ; ...
45      Emma ( 59 ) ; strawberry ( 44 ) ; * book ( 0 )...
54      bear ( 0 ) ; Ella ( 17 ) ; * pencil ( 48 ) ; *...
                              ...
2954    Emma ( 59 ) ; * cake ( 44 ) ; * creature ( 0 )...
2970    * bunny ( 59 ) ; rose ( 44 ) ; Emma ( 0 ) ; li...
2973    Zoe ( 59 ) ; Emma ( 44 ) ; * balloon ( 0 ) ; c...
2988    girl ( 59 ) ; * pillow ( 0 ) ; Brooklyn ( 44 )...
2997    Noah ( 59 ) ; * princess ( 44 ) ; bucket ( 0 )...
```

Figure 1: Examples of errors made by T5-base on the test set. Notice how the predictions include relative positional indices marked by repeating numbers, and how the ground truth outputs do not. This is an example of a common issue with our model, and one that shows up in the CP and PP recursion generalization splits as well.