Meta-Learning to Compositionally Generalize

Henry Conklin[1]*, **Bailin Wang**[1]*, **Kenny Smith**[1] and **Ivan Titov**[1,2]
[1]University of Edinburgh [2]University of Amsterdam
{henry.conklin, bailin.wang, kenny.smith}@ed.ac.uk, ititov@inf.ed.ac.uk

Abstract

Natural language is compositional; the meaning of a sentence is a function of the meaning of its parts. This property allows humans to create and interpret novel sentences, generalizing robustly outside their prior experience. Neural networks have been shown to struggle with this kind of generalization, in particular performing poorly on tasks designed to assess compositional generalization (i.e. where training and testing distributions differ in ways that would be trivial for a compositional strategy to resolve). Their poor performance on these tasks may in part be due to the nature of supervised learning which assumes training and testing data to be drawn from the same distribution. We implement a meta-learning augmented version of supervised learning whose objective directly optimizes for out-of-distribution generalization. We construct pairs of tasks for meta-learning by sub-sampling existing training data. Each pair of tasks is constructed to contain relevant examples, as determined by a similarity metric, in an effort to inhibit models from memorizing their input. Experimental results on the COGS and SCAN datasets show that our similarity-driven meta-learning can improve generalization performance.

1 Introduction

Compositionality is the property of human language that allows for the meaning of a sentence to be constructed from the meaning of its parts and the way in which they are combined (Cann, 1993). By decomposing phrases into known parts we can generalize to novel sentences despite never having encountered them before. In practice this allows us to produce and interpret a functionally limitless number of sentences given finite means (Chomsky, 1965).

Whether or not neural networks can generalize in this way remains unanswered. Prior work asserts that there exist fundamental differences between cognitive and connectionist architectures that makes compositional generalization by the latter unlikely (Fodor and Pylyshyn, 1988). However, recent work has shown these models' capacity for learning some syntactic properties. Hupkes et al. (2018) show how some architectures can handle hierarchy in an algebraic context and generalize in a limited way to unseen depths and lengths. Work looking at the latent representations learned by deep machine translation systems show how these models seem to extract constituency and syntactic class information from data (Blevins et al., 2018; Belinkov et al., 2018). These results, and the more general fact that neural models perform a variety of NLP tasks with high fidelity (eg. Vaswani et al., 2017; Dong and Lapata, 2016), suggest these models have some sensitivity to syntactic structure and by extension may be able to learn to generalize compositionally.

Recently there have been a number of datasets designed to more formally assess connectionist models' aptitude for compositional generalization (Kim and Linzen, 2020; Lake and Baroni, 2018; Hupkes et al., 2019). These datasets frame the problem of compositional generalization as one of out-of-distribution generalization: the model is trained on one distribution and tested on another which differs in ways that would be trivial for a compositional strategy to resolve. A variety of neural network architectures have shown mixed performance across these tasks, failing to show conclusively that connectionist models are reliably capable of generalizing compositionally (Keysers et al., 2020; Lake and Baroni, 2018). Natural language requires a mixture of memorization and generalization (Jiang et al., 2020), memorizing exceptions and atomic concepts with which to generalize. Previous work

*Equal contribution.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 3322–3335
August 1–6, 2021.

looking at compositional generalization has suggested that models may memorize large spans of sentences multiple words in length (Hupkes et al., 2019; Keysers et al., 2020). This practice may not harm in-domain performance, but if at test time the model encounters a sequence of words it has not encountered before it will be unable to interpret it having not learned the atoms (words) that comprise it. Griffiths (2020) looks at the role of limitations in the development of human cognitive mechanisms. Humans' finite computational ability and limited memory may be central to the emergence of robust generalization strategies like compositionality. A hard upper-bound on the amount we can memorize may be in part what forces us to generalize as we do. Without the same restriction models may prefer a strategy that memorizes large sections of the input potentially inhibiting their ability to compositionally generalize.

In a way the difficulty of these models to generalize out of distribution is unsurprising: supervised learning assumes that training and testing data are drawn from the same distribution, and therefore does not necessarily favour strategies that are robust out of distribution. Data necessarily underspecifies for the generalizations that produced it. Accordingly for a given dataset there may be a large number of generalization strategies that are compatible with the data, only some of which will perform well outside of training (D'Amour et al., 2020). It seems connectionist models do not reliably extract the strategies from their training data that generalize well outside of the training distribution. Here we focus on an approach that tries to to introduce a bias during training such that the model arrives at a more robust strategy.

To do this we implement a variant of the model agnostic meta-learning algorithm (MAML, Finn et al., 2017a). The approach used here follows Wang et al. (2020a) which implements an objective function that explicitly optimizes for out-of-distribution generalization in line with Li et al. (2018). Wang et al. (2020a) creates pairs of tasks for each batch (which here we call meta-train and meta-test) by sub-sampling the existing training data. Each meta-train, meta-test task pair is designed to simulate the divergence between training and testing: meta-train is designed to resemble the training distribution, and meta-test to resemble the test distribution. The training objective then requires that update steps taken on meta-train are also beneficial for meta-test. This serves as a kind of regularizer, inhibiting the model from taking update steps that only benefit meta-train. By manipulating the composition of meta-test we can control the nature of the regularization applied. Unlike other meta-learning methods this is not used for few or zero-shot performance. Instead it acts as a kind of meta-augmented supervised learning, that helps the model to generalize robustly outside of its training distribution.

The approach taken by Wang et al. (2020a) relies on the knowledge of the test setting. While it does not assume access to the test distribution, it assumes access to the family of test distributions, from which the actual test distribution will be drawn. While substantially less restrictive than the standard iid setting, it still poses a problem if we do not know the test distribution, or if the model is evaluated in a way that does not lend itself to being represented by discrete pairs of tasks (i.e. if test and train differ in a variety of distinct ways). Here we propose a more general approach that aims to generate meta-train, meta-test pairs which are populated with similar (rather than divergent) examples in an effort to inhibit the model from memorizing its input. Similarity is determined by a string or tree kernel so that for each meta-train task a corresponding meta-test task is created from examples deemed similar.

By selecting for similar examples we design the meta-test task to include examples with many of the same words as meta-train, but in novel combinations. As our training objective encourages gradient steps that are beneficial for both tasks we expect the model to be less likely to memorize large chunks which are unlikely to occur in both tasks, and therefore generalize more compositionally. This generalizes the approach from Wang et al. (2020a), by using the meta-test task to apply a bias not-strictly related to the test distribution: the design of the meta-test task allows us to design the bias which it applies. It is worth noting that other recent approaches to this problem have leveraged data augmentation to make the training distribution more representative of the test distribution (Andreas, 2020). We believe this line of work is orthogonal to ours as it does not focus on getting a model to generalize compositionally, but rather making the task simple enough that compositional generalization is not needed. Our method is model agnostic, and does not require prior knowledge of

the target distribution.

We summarise our contributions as follows:

- We approach the problem of compositional generalization with a meta-learning objective that tries to explicitly reduce input memorization using similarity-driven virtual tasks.
- We perform experiments on two text-to-semantic compositional datasets: COGS and SCAN. Our new training objectives lead to significant improvements in accuracy over a baseline parser trained with conventional supervised learning. [1]

2 Methods

We introduce the meta-learning augmented approach to supervised learning from Li et al. (2018); Wang et al. (2020a) that explicitly optimizes for out-of-distribution generalization. Central to this approach is the generation of tasks for meta-learning by sub-sampling training data. We introduce three kinds of similarity metrics used to guide the construction of these tasks.

2.1 Problem Definition

Compositional Generalization Lake and Baroni (eg. 2018); Kim and Linzen (eg. 2020) introduce datasets designed to assess compositional generalization. These datasets are created by generating synthetic data with different distributions for testing and training. The differences between the distributions are trivially resolved by a compositional strategy. At their core these tasks tend to assess three key components of compositional ability: systematicity, productivity, and primitive application. Systematicity allows for the use of known parts in novel combinations as in (a). Productivity enables generalization to longer sequences than those seen in training as in (b). Primitive application allows for a word only seen in isolation during training to be applied compositionally at test time as in (c).

(a) The cat gives the dog a gift → The dog gives the cat a gift

(b) The cat gives the dog a gift → The cat gives the dog a gift and the bird a gift

(c) made → The cat made the dog a gift

[1] Our implementations are available at https://github.com/berlino/tensor2struct-public.

Algorithm 1 MAML Training Algorithm

Require: Original training set $\mathcal{T}$
Require: Learning rate α, Batch size N
for step $\leftarrow 1$ **to** T **do**
  Sample a random batch from $\mathcal{T}$ as a virtual training set $\mathcal{B}_t$
  Initialize an empty generalization set $\mathcal{B}_g$
  for $i \leftarrow 1$ **to** N **do**
    Sample an example from $\tilde{p}(\cdot \mid \mathcal{B}_t[i])$
    Add it to $\mathcal{B}_g$
  end for
  Construct a virtual task $\tau := (\mathcal{B}_t, \mathcal{B}_g)$
  Meta-train update:
    $\theta' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{B}_t}(\theta)$
  Compute meta-test objective:
    $\mathcal{L}_\tau(\theta) = \mathcal{L}_{\mathcal{B}_t}(\theta) + \mathcal{L}_{\mathcal{B}_g}(\theta')$
  Final Update:
    $\theta \leftarrow \text{Update}(\theta, \nabla_\theta \mathcal{L}_\tau(\theta))$
end for

A compositional grammar like the one that generated the data would be able to resolve these three kinds of generalization easily, and therefore performance on these tasks is taken as an indication of a model's compositional ability.

Conventional Supervised Learning The compositional generalization datasets we look at are semantic parsing tasks, mapping between natural language and a formal representation. A usual supervised learning objective for semantic parsing is to minimize the negative log-likelihood of the correct formal representation given a natural language input sentence, i.e. minimising

$$\mathcal{L}_{\mathcal{B}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(y|x) \quad (1)$$

where N is the size of batch $\mathcal{B}$, y is a formal representation and x is a natural language sentence. This approach assumes that the training and testing data are independent and identically distributed.

Task Distributions Following from Wang et al. (2020a), we utilize a learning algorithm that can enable a parser to benefit from a distribution of virtual tasks, denoted by $p(\tau)$, where τ refers to an instance of a virtual compositional generalization task that has its own training and test examples.

2.2 MAML Training

Once we have constructed our pairs of virtual tasks we need a training algorithm that encourages

compositional generalization in each. Like Wang et al. (2020a), we turn to optimization-based meta-learning algorithms (Finn et al., 2017b; Li et al., 2018) and apply DG-MAML (Domain Generalization with Model-Agnostic Meta-Learning), a variant of MAML (Finn et al., 2017b). Intuitively, DG-MAML encourages optimization on meta-training examples to have a positive effect on the meta-test examples as well.

During each learning episode of MAML training we randomly sample a task τ which consists of a training batch $\mathcal{B}_t$ and a generalization batch $\mathcal{B}_g$ and conduct optimization in two steps, namely *meta-train* and *meta-test*.

Meta-Train The meta-train task is sampled at random from the training data. The model performs one stochastic gradient descent step on this batch

$$\theta' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{B}_t}(\theta) \tag{2}$$

where α is the meta-train learning rate.

Meta-Test The fine-tuned parameters θ' are evaluated on the accompanying generalization task, meta-test, by computing their loss on it denoted as $\mathcal{L}_{\mathcal{B}_g}(\theta')$. The final objective for a task τ is then to jointly optimize the following:

$$\begin{aligned} \mathcal{L}_\tau(\theta) &= \mathcal{L}_{\mathcal{B}_t}(\theta) + \mathcal{L}_{\mathcal{B}_g}(\theta') \\ &= \mathcal{L}_{\mathcal{B}_t}(\theta) + \mathcal{L}_{\mathcal{B}_g}(\theta - \alpha \nabla_\theta \mathcal{L}_\beta(\theta)) \end{aligned} \tag{3}$$

The objective now becomes to reduce the joint loss of both the meta-train and meta-test tasks. Optimizing in this way ensures that updates on meta-train are also beneficial to meta-test. The loss on meta-test acts as a constraint on the loss from meta-train. This is unlike traditional supervised learning ($\mathcal{L}_\tau(\theta) = \mathcal{L}_{\mathcal{B}_t}(\theta) + \mathcal{L}_{\mathcal{B}_g}(\theta)$) where the loss on one batch does not constrain the loss on another.

With a random $\mathcal{B}_t$ and $\mathcal{B}_g$, the joint loss function can be seen as a kind of generic regularizer, ensuring that update steps are not overly beneficial to meta-train alone. By constructing $\mathcal{B}_t$ and $\mathcal{B}_g$ in ways which we expect to be relevant to compositionality, we aim to allow the MAML algorithm to apply specialized regularization during training. Here we design meta-test to be similar to the meta-train task because we believe this highlights the systematicity generalization that is key to compositional ability: selecting for examples comprised of the same atoms but in different arrangements. In constraining each update step with respect to meta-train by performance on similar examples in meta-test we expect the model to dis-prefer a strategy that does not also work for meta-test like memorization of whole phrases or large sections of the input.

Source Example: The girl changed a sandwich beside the table .

Neighbours using Tree Kernel	Similarity
A sandwich changed .	0.55
The girl changed .	0.55
The block was changed by the girl .	0.39
The girl changed the cake .	0.39
change	0.32
Neighbours using String Kernel	
The girl rolled a drink beside the table .	0.35
The girl liked a dealer beside the table .	0.35
The girl cleaned a teacher beside the table .	0.35
The girl froze a bear beside the table .	0.35
The girl grew a pencil beside the table .	0.35
Neighbours using LevDistance	
The girl rolled a drink beside the table .	-2.00
The girl liked a dealer beside the table .	-2.00
The girl cleaned a teacher beside the table .	-2.00
The girl froze a bear beside the table .	-2.00
The girl grew a pencil beside the table .	-2.00

Table 1: Top scoring examples according to the tree kernel, string kernel and Levenshtein distance for the sentence 'The girl changed a sandwich beside the table .' and accompanying scores.

2.3 Similarity Metrics

Ideally, the design of virtual tasks should reflect specific generalization cases for each dataset. However, in practice this requires some prior knowledge of the distribution to which the model will be expected to generalize, which is not always available. Instead we aim to naively structure the virtual tasks to resemble each other. To do this we use a number of similarity measures intended to help select examples which highlight the systematicity of natural language.

Inspired by kernel density estimation (Parzen, 1962), we define a relevance distribution for each example:

$$\tilde{p}(x', y'|x, y) \propto \exp\left(k([x, y], [x', y']/\eta\right) \tag{4}$$

where k is the similarity function, $[x, y]$ is a training example, η is a temperature that controls the sharpness of the distribution. Based on our extended interpretation of relevance, a high $\tilde{p}$ implies that $[x, y]$ is systematically relevant to $[x', y']$ - containing many of the same atoms but in a novel combination. We look at three similarity metrics to guide subsampling existing training data into meta-test tasks proportional to each example's $\tilde{p}$.

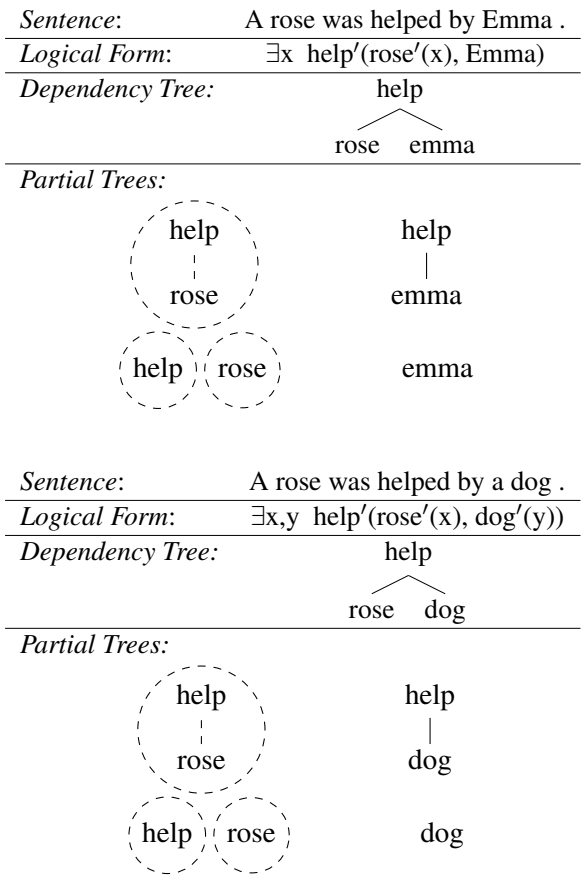


Figure 1: The dependency-tree forms for the logical forms of two sentences. Shown below each tree are its partial trees. As there are three partial trees shared by the examples their un-normalized tree kernel score is 3.

Levenshtein Distance First, we consider Levenshtein distance, a kind of edit distance widely used to measure the dissimilarity between strings. We compute the negative Levenshtein distance at the word-level between natural language sentences of two examples:

$$k([x, y], [x', y']) = -1 * \text{LevDistance}(x, x') \quad (5)$$

where LevDistance returns the number of edit operations required to transform x into x'. See Table 1 for examples.

Another family of similarity metrics for discrete structures are convolution kernels (Haussler, 1999).

String-Kernel Similarity We use the string subsequence kernel (Lodhi et al., 2002):

$$k([x, y], [x', y']) = \text{SSK}(x, x') \quad (6)$$

where SSK computes the number of common subsequences between natural language sentences at the word-level. See Table 1 for examples. [2]

Tree-Kernel Similarity In semantic parsing, the formal representation y usually has a known grammar which can be used to represent it as a tree structure. In light of this we use tree convolution kernels to compute similarity between examples: [3]

$$k([x, y], [x', y']) = \text{TreeKernel}(y, y') \quad (7)$$

where the TreeKernel function is a convolution kernel (Collins and Duffy, 2001) applied to trees. Here we consider a particular case where y is represented as a dependency structure, as shown in Figure 1. We use the partial tree kernel (Moschitti, 2006) which is designed for application to dependency trees. For a given dependency tree partial tree kernels generate a series of all possible partial trees: any set of one or more connected nodes. Given two trees the kernel returns the number of partial trees they have in common, interpreted as a similarity score. Compared with string-based similarity, this kernel prefers sentences that share common syntactic sub-structures, some of which are not assigned high scores in string-based similarity metrics, as shown in Table 1.

Though tree-structured formal representations are more informative in obtaining relevance, not all logical forms can be represented as tree structures. In SCAN (Lake and Baroni, 2018) y are action sequences without given grammars. As we will show in the experiments, string-based similarity metrics have a broader scope of applications but are less effective than tree kernels in cases where y can be tree-structured.

Sampling for Meta-Test Using our kernels we compute the relevance distribution in Eq 4 to construct virtual tasks for MAML training. We show the resulting procedure in Algorithm 1. In order to construct a virtual task τ, a meta-train batch is first sampled at random from the training data (line 2), then the accompanying meta-test batch is created by sampling examples similar to those in meta-train (line 5).

We use *Lev-MAML, Str-MAML and Tree-MAML* to denote the meta-training using Levenshtein distance, string-kernel and tree-kernel similarity, respectively.

[2] We use the normalized convolution kernels in this work, i.e., $k'(x_1, x_2) = k(x_1, x_2)/\sqrt{k(x_1, x_1)k(x_2, x_2)}$

[3] Alternatively, we can use tree edit-distance (Zhang and Shasha, 1989).

3 Experiments

3.1 Datasets and Splits

We evaluate our methods on the following semantic parsing benchmarks that target compositional generalization.

SCAN contains a set of natural language commands and their corresponding action sequences (Lake and Baroni, 2018). We use the Maximum Compound Divergence (MCD) splits (Keysers et al., 2020), which are created based on the principle of maximizing the divergence between the compound (e.g., patterns of 2 or more action sequences) distributions of the training and test tests. We apply Lev-MAML and Str-MAML to SCAN where similarity measures are applied to the natural language commands. Tree-MAML (which uses a tree kernel) is not applied as the action sequences do not have an underlying dependency tree-structure.

COGS contains a diverse set of natural language sentences paired with logical forms based on lambda calculus (Kim and Linzen, 2020). Compared with SCAN, it covers various systematic linguistic abstractions (e.g., passive to active) including examples of lexical and structural generalization, and thus better reflects the compositionality of natural language. In addition to the standard splits of Train/Dev/Test, COGS provides a generalization (Gen) set drawn from a different distribution that specifically assesses compositional generalization. We apply Lev-MAML, Str-MAML and Tree-MAML to COGS; Lev-MAML and Str-MAML make use of the natural language sentences while Tree-MAML uses the dependency structures reconstructed from the logical forms.

3.2 Baselines

In general, our method is model-agnostic and can be coupled with any semantic parser to improve its compositional generalization. Additionally Lev-MAML, and Str-MAML are dataset agnostic provided the dataset has a natural language input. In this work, we apply our methods on two widely used sequence-to-sequences models. [4]

LSTM-based Seq2Seq has been the backbone of many neural semantic parsers (Dong and Lapata, 2016; Jia and Liang, 2016). It utilizes LSTM (Hochreiter and Schmidhuber, 1997) and attention (Bahdanau et al., 2014) under an encoder-decoder (Sutskever et al., 2014) framework.

Transformer-based Seq2Seq also follows the encoder-decoder framework, but it uses Transformers (Vaswani et al., 2017) to replace the LSTM for encoding and decoding. It has proved successful in many NLP tasks e.g., machine translation. Recently, it has been adapted for semantic parsing (Wang et al., 2020b) with superior performance.

We try to see whether our MAML training can improve the compositional generalization of contemporary semantic parsers, compared with standard supervised learning. Moreover, we include a meta-baseline, referred to as Uni-MAML, that constructs meta-train and meta-test splits by uniformly sampling training examples. By comparing with this meta-baseline, we show the effect of similarity-driven construction of meta-learning splits. Note that we do not focus on making comparisons with other methods that feature specialized architectures for SCAN datasets (see Section 5), as these methods do not generalize well to more complex datasets (Furrer et al., 2020).

GECA We additionally apply the good enough compositional augmentation (GECA) method laid out in Andreas (2020) to the SCAN MCD splits. Data augmentation of this kind tries to make the training distribution more representative of the test distribution. This approach is distinct from ours which focuses on the training objective, but the two can be combined with better overall performance as we will show. Specifically, we show the results of GECA applied to the MCD splits as well as GECA combined with our Lev-MAML variant. Note that we elect not to apply GECA to COGS, as the time and space complexity [5] of GECA proves very costly for COGS in our preliminary experiments.

3.3 Construction of Virtual Tasks

The similarity-driven sampling distribution $\tilde{p}$ in Eq 4 requires computing the similarity between every pair of training examples, which can be very expensive depending on the size of of the dataset. As the sampling distributions are fixed during training, we compute and cache them beforehand. However, they take an excess of disk space to store as essentially we need to store an $N \times N$ matrix where N

[4] Details of implementations and hyperparameters can be found in the Appendix.

[5] See the original paper for details.

Model	MCD1	MCD2	MCD3
LSTM	4.7 ±2.2	7.3 ±2.1	1.8 ±0.7
Transformer	0.4 ±0.4	1.8 ±0.4	0.5 ±0.1
T5-base	26.2 ±1.7	7.9 ±1.6	12.1 ±0.1
T5-11B	7.9	2.4	**16.8**
LSTM	27.4 ±8.2	31.0 ±0.4	9.6 ±3.7
w. Uni-MAML	44.8 ±5.4	31.9 ±3.4	10.0 ±1.4
w. Lev-MAML	**47.6** ±2.3	**35.2** ±3.9	11.4 ±3.0
w. Str-MAML	42.2 ±2.6	33.6 ±4.3	11.4 ±2.2
Transformer	2.6 ±0.8	3.1 ±1.0	2.3 ±1.3
w. Uni-MAML	2.8 ±0.7	3.2 ±1.0	3.2 ±1.6
w. Lev-MAML	4.7 ±1.8	6.7 ±1.4	6.5 ±1.2
w. Str-MAML	2.8 ±0.6	5.6 ±1.6	6.7 ±1.4
GECA + LSTM	51.5 ±4.4	30.4 ±4.8	12.0 ±6.8
w. Lev-MAML	**58.9** ±6.4	**34.5** ±2.5	12.3 ±4.9

Table 2: Main results on SCAN MCD splits. We show the mean and variance (95% confidence interval) of 10 runs. Cells with a grey background are results obtained in this paper, whereas cells with a white background are from Furrer et al. (2020).

Model	Gen Dev	Test	Gen
LSTM	-	99	16 ±8
Transformer	-	96	35 ±6
LSTM	30.3 ±7.3	99.7	34.5 ±4.5
w. Uni-MAML	36.1 ±6.7	99.7	36.4 ±3.6
w. Lev-MAML	35.6 ±5.3	99.7	36.4 ±5.2
w. Str-MAML	36.3 ±4.2	99.7	36.8 ±3.5
w. Tree-MAML	**41.2** ±2.8	99.7	**41.0** ±4.9
Transformer	54.7 ±4.0	99.5	58.6 ±3.7
w. Uni-MAML	60.9 ±2.8	99.6	64.4 ±4.0
w. Lev-MAML	62.7 ±3.8	99.7	64.9 ±6.3
w. Str-MAML	62.3 ±3.0	99.6	64.8 ±5.5
w. Tree-MAML	**64.1** ±3.2	99.6	**66.7** ±4.4

Table 3: Main results on the COGS dataset. We show the mean and variance (standard deviation) of 10 runs. Cells with a grey background are results obtained in this paper, whereas cells with a white background are from Kim and Linzen (2020).

is the number of training examples. To allow efficient storage and sampling, we use the following approximation. First, we found that usually each example only has a small set of neighbours that are relevant to it. [6] Motivated by this observation, we only store the top 1000 relevant neighbours for each example sorted by similarity, and use it to construct the sampling distribution denoted as $\tilde{p}_{\text{top1000}}$. To allow examples out of top 1000 being sampled, we use a linear interpolation between $\tilde{p}_{\text{top1000}}$ and a uniform distribution. Specifically, we end up using the following sampling distribution:

$$\tilde{p}(x', y'|x, y) = \lambda\, \tilde{p}_{\text{top1000}}(x', y'|x, y) + (1-\lambda)\frac{1}{N}$$

where $\tilde{p}_{\text{top1000}}$ assigns 0 probability to out-of top 1000 examples, N is the number of training examples, and λ is a hyperparameter for interpolation. In practice, we set λ to 0.5 in all experiments. To sample from this distribution, we first decide whether the sample is in the top 1000 by sampling from a Bernoulli distribution parameterized by λ. If it is, we use $\tilde{p}_{\text{top1000}}$ to do the sampling; otherwise, we uniformly sample an example from the training set.

3.4 Development Set

Many tasks that assess out-of-distribution (O.O.D.) generalization (e.g. COGS) do not have an O.O.D. Dev set that is representative of the generalization distribution. This is desirable as a parser in principle should never have knowledge of the Gen set during training. In practice though the lack of an O.O.D. Dev set makes model selection extremely difficult and not reproducible. [7] In this work, we propose the following strategy to alleviate this issue: 1) we sample a small subset from the Gen set, denoted as 'Gen Dev' for tuning meta-learning hyperparmeters, 2) we use two disjoint sets of random seeds for development and testing respectively, i.e., retraining the selected models from scratch before applying them to the final test set. In this way, we make sure that our tuning is not exploiting the models resulting from specific random seeds: we do not perform random seed tuning. At no point are any of our models trained on the Gen Dev set.

3.5 Main Results

On SCAN, as shown in Table 2, Lev-MAML substantially helps both base parsers achieve better performance across three different splits constructed according to the MCD principle. [8] Though our models do not utilize pre-training such as T5 (Raffel et al., 2019), our best model (Lev-MAML + LSTM) still outperforms T5 based models significantly in MCD1 and MCD2. We show that GECA is also effective for MCD splits (especially

[6] For example, in COGS, each example only retrieves 3.6% of the whole training set as its neighbours (i.e., have non-zero tree-kernel similarity) on average.

[7] We elaborate on this issue in the Appendix.

[8] Our base parsers also perform much better than previous methods, likely due to the choice of hyperparameters.

in MCD1). More importantly, augmenting GECA with Lev-MAML further boosts the performance substantially in MCD1 and MCD2, signifying that our MAML training is complementary to GECA to some degree.

Table 3 shows our results on COGS. Tree-MAML boosts the performance of both LSTM and Transformer base parsers by a large margin: 6.5% and 8.1% respectively in average accuracy. Moreover, Tree-MAML is consistently better than other MAML variants, showing the effectiveness of exploiting tree structures of formal representation to construct virtual tasks. [9]

4 Discussion

4.1 SCAN Discussion

The application of our string-similarity driven meta-learning approaches to the SCAN dataset improved the performance of the LSTM baseline parser. Our results are reported on three splits of the dataset generated according to the maximum compound divergence (MCD) principle. We report results on the only MCD tasks for SCAN as these tasks explicitly focus on the systematicity of language. As such they assess a model's ability to extract sufficiently atomic concepts from its input, such that it can still recognize those concepts in a new context (i.e. as part of a different compound). To succeed here a model must learn atoms from the training data and apply them compositionally at test time. The improvement in performance our approach achieves on this task suggests that it does disincentivise the model from memorizing large sections - or entire compounds - from its input.

GECA applied to the SCAN MCD splits does improve performance of the baseline, however not to the same extent as when applied to other SCAN tasks in Andreas (2020). GECA's improvement is comparable to our meta-learning method, despite the fact that our method does not leverage any data augmentation. This means that our method achieves high performance by generalizing robustly outside of its training distribution, rather than by making its training data more representative of the test distribution. The application of our Lev-MAML approach to GECA-augmented data results in further improvements in performance, suggesting that these approaches aid the model in distinct yet complementary ways.

4.2 COGS Discussion

All variants of our meta-learning approach improved both the LSTM and Transformer baseline parsers' performance on the COGS dataset. The Tree-MAML method outperforms the Lev-MAML, Str-MAML, and Uni-MAML versions. The only difference between these methods is the similarity metric used, and so differences in performance must be driven by what each metric selects for. For further analysis of the metrics refer to the appendix.

The strong performance of the Uni-MAML variant highlights the usefulness of our approach generally in improving models' generalization performance. Even without a specially designed meta-test task this approach substantially improves on the baseline Transformer model. We see this as evidence that this kind of meta-augmented supervised learning acts as a robust regularizer particularly for tasks requiring out of distribution generalization.

Although the Uni-MAML, Lev-MAML, and Str-MAML versions perform similarly overall on the COGS dataset they may select for different generalization strategies. The COGS generalization set is comprised of 21 sub-tasks which can be used to better understand the ways in which a model is generalizing (refer to Table 4 for examples of subtask performance). Despite having very similar overall performance Uni-MAML and Str-MAML perform distinctly on individual COGS tasks - with their performance appearing to diverge on a number of of them. This would suggest that the design of the meta-test task may have a substantive impact on the kind of generalization strategy that emerges in the model. For further analysis of COGS sub-task performance see the appendix.

Our approaches' strong results on both of these datasets suggest that it aids compositional generalization generally. However it is worth nothing that both datasets shown here are synthetic, and although COGS endeavours to be similar to natural data, the application of our methods outside of synthetic datasets is important future work.

5 Related Work

Compositional Generalization A large body of work on compositional generalization provide models with strong compositional bias, such as specialized neural architectures (Li et al., 2019; Russin

[9] The improvement of all of our MAML variants applied to the Transformer are significant ($p < 0.03$) compared to the baseline, of our methods applied to LSTMs, Tree-MAML is significant ($p < 0.01$) compared to the baseline.

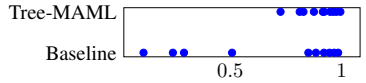
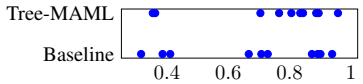
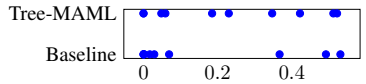
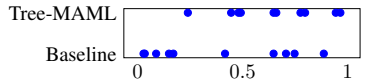
Case	Training	Generalization	Accuracy Distribution
Primitive noun → Subject (common noun)	**shark**	A **shark** examined the child.	
Primitive noun → Subject (proper noun)	**Paula**	**Paula** sketched William.	
Primitive noun → Object (common noun)	**shark**	A chief heard the **shark**.	
Primitive noun → Object (proper noun)	**Paula**	The child helped **Paula**.	

Table 4: Accuracy on COGS by generalization case. Each dot represents a single run of the model.

et al., 2019; Gordon et al., 2019), or grammar-based models that accommodate alignments between natural language utterances and programs (Shaw et al., 2020; Herzig and Berant, 2020). Another line of work utilizes data augmentation via fixed rules (Andreas, 2020) or a learned network (Akyürek et al., 2020) in an effort to transform the out-of-distribution compositional generalization task into an in-distribution one. Our work follows an orthogonal direction, injecting compositional bias using a specialized training algorithm. A related area of research looks at the emergence of compositional languages, often showing that languages which seem to lack natural-language like compositional structure may still be able to generalize to novel concepts (Kottur et al., 2017; Chaabouni et al., 2020). This may help to explain the ways in which models can generalize robustly on in-distribution data unseen during training while still struggling on tasks specifically targeting compositionality.

Meta-Learning for NLP Meta-learning methods (Vinyals et al., 2016; Ravi and Larochelle, 2016; Finn et al., 2017b) that are widely used for few-shot learning, have been adapted for NLP applications like machine translation (Gu et al., 2018) and relation classification (Obamuyide and Vlachos, 2019). In this work, we extend the conventional MAML (Finn et al., 2017b) algorithm, which was initially proposed for few-shot learning, as a tool to inject inductive bias, inspired by Li et al. (2018); Wang et al. (2020a). For compositional generalization, Lake (2019) proposes a meta-learning procedure to train a memory-augmented neural model. However, its meta-learning algorithm is specialized for the SCAN dataset (Lake and Baroni, 2018) and not suitable to more realistic datasets.

6 Conclusion

Our work highlights the importance of training objectives that select for robust generalization strategies. The meta-learning augmented approach to supervised learning used here allows for the specification of different constraints on learning through the design of the meta-tasks. Our similarity-driven task design improved on baseline performance on two different compositional generalization datasets, by inhibiting the model's ability to memorize large sections of its input. Importantly though the overall approach used here is model agnostic, with portions of it (Str-MAML, Lev-MAML, and Uni-MAML) proving dataset agnostic as well requiring only that the input be a natural language sentence. Our methods are simple to implement compared with other approaches to improving compositional generalization, and we look forward to their use in combination with other techniques to further improve models' compositional ability.

Acknowledgements

This work was supported in part by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) and the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences. We also acknowledge the financial support of the European Research Council (Titov, ERC StG BroadSem 678254) and the Dutch National Science Foundation (Titov, NWO VIDI 639.022.518).

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2020. Learning to recombine and resam-

ple data for compositional generalization. *arXiv preprint arXiv:2010.03706*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating Layers of Representation in Neural Machine Translation on Part-of-Speech and Semantic Tagging Tasks. *arXiv:1801.07772 [cs]*. ArXiv: 1801.07772.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. *arXiv:1805.04218 [cs]*. ArXiv: 1805.04218.

Ronnie Cann. 1993. *Formal semantics an introduction*. Cambridge University Press, Cambridge [etc. OCLC: 1120437841.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. 2020. Compositionality and generalization in emergent languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4427–4442, Online. Association for Computational Linguistics.

Noam Chomsky. 1965. *Aspects of the theory of syntax*, 50th anniversary edition edition. Number no. 11 in Massachusetts Institute of Technology. Research Laboratory of Electronics. Special technical report. The MIT Press, Cambridge, Massachusetts.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in neural information processing systems*, pages 625–632.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. 2020. Underspecification Presents Challenges for Credibility in Modern Machine Learning. *arXiv:2011.03395 [cs, stat]*. ArXiv: 2011.03395.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017a. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]*. ArXiv: 1703.03400.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017b. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

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

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

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2019. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Thomas L Griffiths. 2020. Understanding human intelligence through human limitations. *Trends in Cognitive Sciences*.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

David Haussler. 1999. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California ....

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040*.

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

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2019. The compositionality of neural networks: integrating symbolism and connectionism. *arXiv:1908.08351 [cs, stat]*. ArXiv: 1908.08351.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C. Mozer. 2020. Characterizing Structural Regularities of Labeled Data in Overparameterized Models. *arXiv:2002.03206 [cs, stat]*. ArXiv: 2002.03206.

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*.

Najoung Kim and Tal Linzen. 2020. COGS: A Compositional Generalization Challenge Based on Semantic Interpretation. *arXiv:2010.05465 [cs]*. ArXiv: 2010.05465.

Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Natural language does not emerge'naturally'in multi-agent dialog. *arXiv preprint arXiv:1706.08502*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. *arXiv preprint arXiv:1906.05381*.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. 2018. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. *arXiv preprint arXiv:1910.02612*.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*, pages 318–329. Springer.

Abiola Obamuyide and Andreas Vlachos. 2019. Model-agnostic meta-learning for relation classification with limited supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5873–5879, Florence, Italy. Association for Computational Linguistics.

Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning.

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2020. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? *arXiv preprint arXiv:2010.12725*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2020a. Meta-learning for domain generalization in semantic parsing. *arXiv preprint arXiv:2010.11988*.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020b. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

A Experiments

A.1 Details of Base Parsers

We implemented all models with Pytorch (Paszke et al., 2019). For the LSTM parsers, we use a two-layer encoder and one-layer decoder with attention (Bahdanau et al., 2014) and input-feeding (Luong et al., 2015). We only test bidirectional LSTM encoders, as unidirectional LSTM models do not perform very well in our preliminary experiments. For Transformer parsers, we use 2 encoder and decoder layers, 4 attention heads, and a feed-forward dimension of 1024. The hidden size for both LSTM and Transformer models are 256. The hyparameters of base parsers are mostly borrowed from related work and not tuned, as the primary goal of this work is the MAML training algorithm. To experiment with a wide variety of possible Seq2Seq models, we also try a Transformer encoder + LSTM decoder and find that this variant actually performs slightly better than both vanilla Transformer and LSTM models. Further exploration of this combination in pursuit of a better neural architecture for compositional generalization might be interesting for future work.

A.2 Model Selection Protocol

In our preliminary experiments on COGS, we find almost all the Seq2Seq models achieve $> 99\%$ in accuracy on the original Dev set. However, their performance on the Gen set diverge dramatically, ranging from 10% to 70%. The lack of an informative Dev set makes model selection extremely difficult and difficult to reproduce. This issue might also be one of the factors that results in the large variance of performance reported in previous work. Meanwhile, we found that some random seeds [10] yield consistently better performance than others across different conditions. For example, among the ten random seeds used for Lev-MAML + Transformer on COGS, the best performing seed obtains 73% whereas the lowest performing seed obtains 54%. Thus, it is important to compare different models using the same set of random seeds, and not to tune the random seeds in any model. To alleviate these two concerns, we choose the protocol that is mentioned in the main paper. This protocol helps to make the results reported in our paper reproducible.

[10] Random seeds control the initialization of parameters and the order of training batches.

A.3 Details of Training and Evaluation

Following Kim and Linzen (2020), we train all models from scratch using randomly initialized embeddings. For SCAN, models are trained for 1,000 steps with batch size 128. We choose model checkpoints based on their performance on the Dev set. For COGS, models are trained for 6,000 steps with batch size of 128. We choose the meta-train learning rate α in Equation 2, temperature η in Equation 4 based on the performance on the Gen Dev set. Finally we use the chosen α, η to train models with new random seeds, and only the last checkpoints (at step 6,000) are used for evaluation on the Test and Gen set.

A.4 Other Splits of SCAN

The SCAN dataset contains many splits, such as Add-Jump, Around Right, and Length split, each assessing a particular case of compositional generalization. We think that MCD splits are more representative of compositional generalization due to the nature of the principle of maximum compound divergence. Moreover, it is more challenging than other splits (except the Length split) according to Furrer et al. (2020). That GECA, which obtains 82% in accuracy on JUMP and Around Right splits, only obtains $< 52\%$ in accuracy on MCD splits in our experiments confirms that MCD splits are more challenging.

A.5 Kernel Analysis

The primary difference between the tree-kernel and string-kernel methods is in the diversity of the examples they select for the meta-test task. The tree kernel selects a broader range of lengths, often including atomic examples, a single word in length, matching a word in the original example from meta-train (see table 5). By design the partial tree kernel will always assign a non-zero value to an example that is an atom contained in the original sentence. We believe the diversity of the sentences selected

Partial Tree Kernel	**top 10**	**100**	**1000**	**LevDistance**	**top 10**	**100**	**1000**
Mean Example Length (chars)	26.71	26.59	29.87	Mean Example Length (chars)	31.04	30.45	29.28
Std dev	± 6.80	± 7.61	± 8.85	Std dev	± 2.80	± 3.77	± 4.78
Mean No. of Atoms	0.46	0.81	1.13	Mean No. of Atoms	0.00	0.00	0.02
Std dev	± 0.67	± 1.05	± 0.81	Std dev	± 0.00	± 0.02	± 0.17

Table 5: Analyses of kernel diversity. Reporting mean example length and number of atoms for the top k highest scoring examples for each kernel. Note that atoms are only counted that also occur in the original example.

Source Example: Emma lended the donut to the dog .

Neighbours using Tree Kernel	Similarity
Emma was lended the donut .	0.74
The donut was lended to Emma .	0.62
Emma lended the donut to a dog .	0.55
Emma lended Liam the donut .	0.55
Emma lended a girl the donut .	0.55
Neighbours using String Kernel	
Emma lended the donut to a dog .	0.61
Emma lended the box to a dog .	0.36
Emma gave the cake to the dog .	0.33
Emma lended the cake to the girl .	0.33
Emma lended the liver to the girl .	0.33
Neighbours using LevDistance	
Emma lended the donut to a dog .	-1.00
Emma loaned the donut to the teacher .	-2.00
Emma forwarded the donut to the monster .	-2.00
Emma gave the cake to the dog .	-2.00
Charlotte lended the donut to the fish .	-2.00

Source Example: The crocodile valued that a girl snapped .

Neighbours using Tree Kernel	Similarity
A girl snapped .	0.55
A rose was snapped by a girl .	0.39
The cookie was snapped by a girl .	0.39
girl	0.32
value	0.32
Neighbours using String Kernel	
The crocodile liked a girl .	0.28
The girl snapped .	0.27
The crocodile hoped that a boy observed a girl .	0.26
The boy hoped that a girl juggled .	0.15
The cat hoped that a girl sketched .	0.15
Neighbours using LevDistance	
The crocodile liked a girl .	-3.00
The boy hoped that a girl juggled .	-3.00
The cat hoped that a girl sketched .	-3.00
The cat hoped that a girl smiled .	-3.00
Emma liked that a girl saw .	-4.00

Table 6: Top scoring examples according to the tree kernel, string kernel and Levenshtein distance for two sentences and accompanying scores.

by the tree kernel accounts for the superior performance of Tree-MAML compared with the other MAML conditions. The selection of a variety of lengths for meta-test constrains model updates on the meta-train task such that they must also accommodate the diverse and often atomic examples selected for meta-test. This constraint would seem to better inhibit memorizing large spans of the input unlikely to be present in meta-test.

A.6 Meta-Test Examples

In Table 6, we show top scoring examples retrieved by the similarity metrics for two sentences. We found that in some cases (e.g., the right part of Table 6), the tree-kernel can retrieve examples that diverge in length but are still semantically relevant. In contrast, string-based similarity metrics, especially LevDistance, tends to choose examples with similar lengths.

A.7 COGS Subtask Analysis

We notice distinct performance for different conditions on the different subtasks from the COGS dataset. In Figure 2 we show the performance of the Uni-MAML and Str-MAML conditions compared with the mean of those conditions. Where the bars are equal to zero the models' performance on that task is roughly equal.

Full task names for figure 2:
(1) prim→subj proper,
(2) active→passive,
(3) only seen as unacc subj → unerg subj,
(4) subj→obj proper,
(5) only seen as unacc subj → obj omitted transitive subj,
(6) pp recursion,
(7) cp recursion,
(8) obj pp→subj pp,
(9) obj→subj common,
(10) do dative→pp dative,
(11) passive→active,

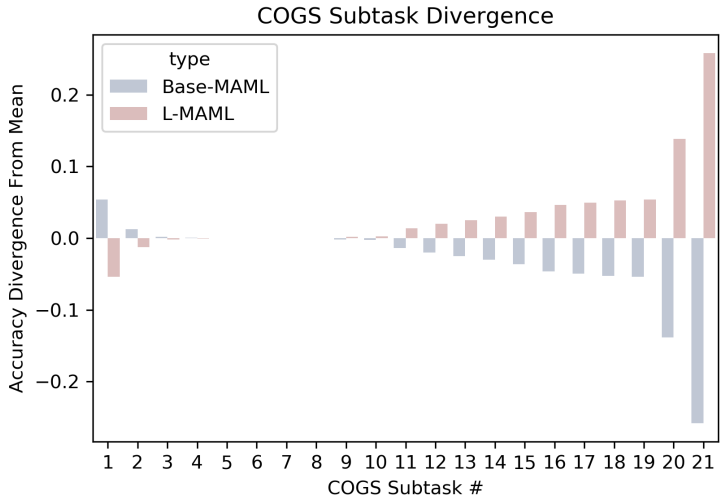


Figure 2: Performance for the Uni-MAML and Lev-MAML conditions compared to the mean of those two conditions.

(12) only seen as transitive subj → unacc subj,
(13) obj omitted transitive→transitive,
(14) subj→obj common,
(15) prim→obj proper,
(16) obj→subj proper,
(17) pp dative→do dative,
(18) unacc→transitive,
(19) prim→subj common,
(20) prim→obj common,
(21) prim→inf arg.