Compositional Generalisation with Structured Reordering and Fertility Layers

Matthias Lindemann[1] and **Alexander Koller**[2] and **Ivan Titov**[1,3]
[1] ILCC, University of Edinburgh, [2] LST, Saarland University, [3] ILLC, University of Amsterdam
m.m.lindemann@sms.ed.ac.uk, koller@coli.uni-saarland.de, ititov@inf.ed.ac.uk

Abstract

Seq2seq models have been shown to struggle with compositional generalisation, i.e. generalising to new and potentially more complex structures than seen during training. Taking inspiration from grammar-based models that excel at compositional generalisation, we present a flexible end-to-end differentiable neural model that composes two structural operations: a fertility step, which we introduce in this work, and a reordering step based on previous work (Wang et al., 2021). To ensure differentiability, we use the expected value of each step, which we compute using dynamic programming. Our model outperforms seq2seq models by a wide margin on challenging compositional splits of realistic semantic parsing tasks that require generalisation to longer examples. It also compares favourably to other models targeting compositional generalisation.[1]

1 Introduction

Many NLP tasks require translating an input object such as a sentence into a structured output object such as a semantic parse. Recently, these tasks have been approached with seq2seq models with great success. However, seq2seq models also have been shown to struggle *out-of-distribution* on compositional generalisation (Lake and Baroni, 2018; Finegan-Dollak et al., 2018; Kim and Linzen, 2020; Hupkes et al., 2020), i.e. the model fails on examples that contain unseen compositions or deeper recursion of phenomena that it handles correctly in isolation.

Consider the example in Fig. 1. Arguably, any model that produces the given semantic parse from the input in a generalisable way has to capture the correspondence between fragments of the input and fragments of the output, at least implicitly. This is challenging because of structural mismatches between input and output. For example, the fragment contributed by “flights" is discontinuous and intertwined with the rest of the semantic parse.

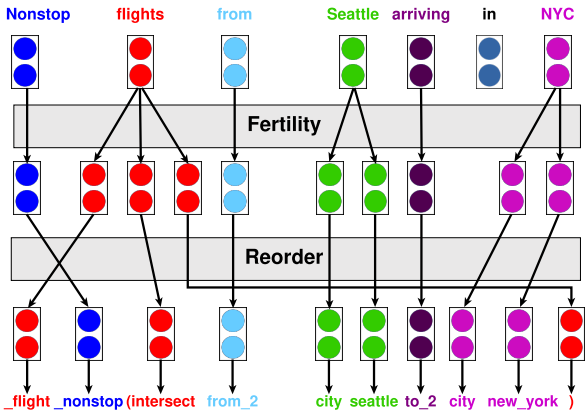

Figure 1: We model structural seq2seq tasks as the composition of differentiable fertility and phrase reordering layers. The model is trained end-to-end without direct supervision of the two structural layers.

In contrast to seq2seq models, grammar-based models such as synchronous context-free grammars (SCFGs) (Lewis and Stearns, 1968; Chiang, 2007) explicitly capture the compositional process behind the data and therefore perform very well in compositional generalisation setups. They can also model common structural mismatches like the one shown in the example. However, grammar-based models are rigid and brittle and thus do not scale well.

In this work, we take inspiration from grammar-based models and present an end-to-end differentiable neural model that is both flexible and generalises well compositionally. Our model consists of two structural layers: a phrase reordering layer originally introduced by Wang et al. (2021) and a fertility layer, new in this work, which creates zero or more *copies* of the representation of any input token. We show how to compose these layers to achieve strong generalisation.

We use a simple decoder and essentially translate each token after the fertility and reordering layers

[1] https://github.com/namednil/f-then-r

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2172–2186
May 2-6, 2023

independently into one output token. We found this to lead to better generalisation than using an LSTM-based decoder. The simple decoder also makes it easy to integrate grammar constraints to ensure well-formed outputs, e.g. in semantic parsing.

Most seq2seq models tend to predict the end of the sequence too early on instances that are longer than seen in training (Newman et al., 2020). Our model overcomes this by explicitly predicting the length of the output sequence as a sum of fertilities.

We first demonstrate the expressive capacity and inductive bias of our model on synthetic data. We then evaluate on three English semantic parsing datasets (Geoquery, ATIS, Okapi) and a style transfer task (Lyu et al., 2021). We clearly outperform seq2seq models both with and without pretraining in structural generalisation setups, particularly when the model has to generalise to longer examples. Our model also compares favourably with existing approaches that target structural generalisation, and we obtain state-of-the-art results on the structural style transfer tasks.

To summarise, our main contributions are:

- an efficient differentiable fertility layer;
- a flexible end-to-end differentiable model that composes two structural operations (fertility and reordering) and achieves strong performance in structural generalisation tasks.

2 Related Work

Fertility The concept of fertility was introduced by Brown et al. (1990) for statistical machine translation to capture that a word in one language is often consistently translated to a certain number of words in another language. Tu et al. (2016) and Malaviya et al. (2018) incorporate fertility into the attention mechanism of seq2seq models. Cohn et al. (2016); Gu et al. (2016) use heuristic supervision for training their fertility models. In contrast to prior work, we learn an explicit fertility component jointly with the rest of our model.

Monotonic alignment Related to fertility is the concept of monotonic alignment, i.e. an alignment a that maps output positions to input positions such that for any two output positions $i < j$, $a(i) \leq a(j)$. Monotonic alignments are usually modelled by an HMM-like model that places the monotonicity constraint on the transition matrix (Yu et al., 2016; Wu and Cotterell, 2019), leading to a runtime of $O(|\mathbf{x}|^2|\mathbf{y}|)$ with $\mathbf{x}$ being the input and $\mathbf{y}$ the output. Raffel et al. (2017) parameterise the alignment using a series of Bernoulli trials and obtain a training runtime of $O(|\mathbf{x}||\mathbf{y}|)$. Our approach also has $O(|\mathbf{x}||\mathbf{y}|)$ runtime.

Compositional generalisation There is a growing body of work on improving the ability of neural models to generalise compositionally in semantic parsing. Good progress has been made in terms of generalisation to new lexical items (Andreas, 2020; Akyürek and Andreas, 2021; Conklin et al., 2021; Csordás et al., 2021; Ontañón et al., 2022) but structural generalisation remains very challenging (Oren et al., 2020; Bogin et al., 2022).

Herzig and Berant (2021) use a neural chart parser and induce latent trees with an approach similar to hard EM. Their model assumes that one input token corresponds to a single leaf in the tree. Zheng and Lapata (2022) re-encode the input and partially generated output with a transformer for every decoding step to reduce entanglement and show considerable gains in structural generalisation.

There has also been work inspired by quasi-synchronous grammars (QCFGs, Smith and Eisner (2006)). Shaw et al. (2021) heuristically induce a QCFG and create an ensemble of a QCFG-based parser and a seq2seq model. Qiu et al. (2021) use similar QCFGs for data augmentation for a seq2seq model. Our approach does not require constructing a grammar. Finally, Kim (2021) introduces neural QCFGs which perform well on compositional generalisation tasks but are very compute-intensive.

Closest to our work is that of Wang et al. (2021) who reorder phrases and use a monotonic attention mechanism on top. Our approach differs from theirs in several important aspects: (i) we use fertility instead of monotonic attention, which parameterises alignments differently; (ii) we apply the fertility step first and *then* reorder the phrases, so our model can directly create alignments where output tokens aligned to the same input token do not have to be consecutive; (iii) we predict the length as the sum of the fertilities and not with an end-of-sequence token; (iv) they use an LSTM-based decoder network whereas we found that a simpler decoder can generalise better.

3 Background

3.1 Structured Attention

We often want to model the relationship between input $\mathbf{x}$ of length n and output $\mathbf{y}$ of length l by

means of a latent variable Z. In particular, we assume that $Z \in \mathcal{Z} \subseteq \mathbb{B}^{n \times l}$ is a boolean alignment matrix. We want to predict $\mathbf{y}$ from a representation produced by a function $g(Z, \mathbf{x})$. In this case, the marginal distribution

$$P(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{P(Z|\mathbf{x})} P(\mathbf{y}|g(Z, \mathbf{x})) \quad (1)$$

can be intractable to compute because $\mathcal{Z}$ often has exponential size in $\mathbf{x}$. In some important cases however we can formulate a similar but tractable model by using *structured attention* (Kim et al., 2017) which 'pushes' the expectation inside the model:

$$P(\mathbf{y}|\mathbf{x}) \approx P(\mathbf{y}|g(\tilde{Z}, \mathbf{x}))$$

where $\tilde{Z} = \mathbb{E}_{P(Z|\mathbf{x})} Z$ is now a 'soft' rather than a boolean matrix. Note that $\tilde{Z}_{ij} = P(Z_{ij} = 1|\mathbf{x})$ is a marginal probability that often can be efficiently computed with a dynamic programme if $P(Z|\mathbf{x})$ is factorisable. We will use such an approach for our model.

3.2 Marginal Permutations

In this section, we briefly review the method of Wang et al. (2021) that we use in our model.

Wang et al. (2021) build on bracketing transduction grammars (Wu, 1997) and show how to compute a distribution over separable permutations (Bose et al., 1998). A permutation is separable, iff it can be represented as a permutation tree. Some permutations are not separable. The internal nodes of a permutation tree are labelled as $\wedge$ or $\triangle$ and are interpreted as operations: $\wedge$ concatenates the values it receives from its left child with the value from its right child, whereas $\triangle$ concatenates them in reverse order. For example, the permutation tree $t = (\triangle\,(\wedge\,a\;b)\,(\triangle\,c\;d))$ represents the permutation $abcd \rightarrow dcab$. Let $R_t(i, j) = 1$ if the permutation described by tree t maps position i to position j, otherwise $R_t(i, j) = 0$.

Wang et al. (2021) show how to compute the expected permutation matrix $\tilde{R}_{i,j} \triangleq \mathbb{E}_{P(t|\mathbf{x})} R_t(i, j)$ in polynomial time with a CYK-style algorithm if $P(t|\mathbf{x})$ factors according to the CYK chart. Crucially, the expected permutation can also be interpreted as a distribution over alignments, where $\tilde{R}_{i,j}$ is the *marginal probability* that position i aligns to position j. We will use this alignment distribution as a building block in our model.

4 Overview of the Approach

In this section, we give a general overview of our approach and defer the details to Sections 5 and 6.

Conceptually, we want to model the transduction from input $\mathbf{x}$ of length n into output $\mathbf{y}$ of length l as the composition of two edit operations. First, we apply a *fertility* step, in which we decide for each token what its fertility is, i.e. how many *copies* we make of it. Assigning fertility of 0 corresponds to deleting the token. This step yields an *intermediate* sequence of tokens. We then reorder them using permutation trees (see Section 3.2). In the last step, we individually translate these tokens into the output tokens. Fig. 1 shows an example where the fertility step and reordering apply to vector representations of tokens.

The fertility step and the reordering can be represented as boolean matrices $F \in \mathbb{B}^{n \times l}$ and $R \in \mathbb{B}^{l \times l}$, respectively. These matrices denote alignments between the sequences before and after the operation. For example, $F_{i,j} = 1$ means that input token i aligns to intermediate token j, i.e. j is one of the copies of i.

With this conceptualisation, we would ideally use the following probabilistic model $P(\mathbf{y}|\mathbf{x})$:

$$P(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\underbrace{P(F|\mathbf{x})}_{\text{Fertility}}} [\mathbb{E}_{\underbrace{P(R|\mathbf{x}, F)}_{\text{Reordering}}} \underbrace{P(\mathbf{y}|\mathbf{x}, F, R)}_{\text{Decoder}}]$$

At training time, the true fertility values are unknown but we observe the length l of $\mathbf{y}$, so we condition on it:

$$P(\mathbf{y}|\mathbf{x}) = P(l|\mathbf{x}) \cdot \mathbb{E}_{P(F,R|\mathbf{x},l)} P(\mathbf{y}|\mathbf{x}, F, R) \quad (2)$$

where $P(l|\mathbf{x})$ can be computed with dynamic programming relying on the fertility model, as we explain in the next section.

Computing the marginal likelihood and the gradients is intractable. Instead of computing the likelihood exactly, one could sample and use a score function estimator (Williams, 1992) but the resulting gradient estimates have high variance.

Instead, we use structured attention as discussed in Section 3.1 and 'push' the expectations inside the model:

$$\mathbb{E}_{P(F,R|\mathbf{x},l)} P(\mathbf{y}|\mathbf{x}, F, R) \approx P(\mathbf{y}|\mathbf{x}, \tilde{F}, \tilde{R}) \quad (3)$$

with $\tilde{F} = \mathbb{E}_{P(F|\mathbf{x},l)} F$ and $\tilde{R} = \mathbb{E}_{P(R|\mathbf{x},\tilde{F})} R$. $\tilde{F}$ and $\tilde{R}$ now represent 'soft', differentiable versions of fertility and reordering. They approach their

discrete counterparts as $P(F|\mathbf{x}, l)$ and $P(R|\tilde{F}, \mathbf{x})$ become peakier, which tends to happen over the course of training. We can view $(\tilde{F}\tilde{R})_{i,j} = \sum_k \tilde{F}_{i,k}\tilde{R}_{k,j}$ as a probability of aligning i to j in the composition of the operations.

A tendency to memorise larger chunks of the output might contribute to poor compositional generalisation (Hupkes et al., 2020). In order to avoid this, we use a simple decoder that generates each output token independently. A first attempt might look like this:

$$P(\mathbf{y}|\mathbf{x}, \tilde{F}, \tilde{R}) = \prod_{i=1}^{l} \sum_{j,k} P(\mathbf{y}_i|\mathbf{x}_j)\tilde{F}_{j,k}\tilde{R}_{k,i} \quad (4)$$

where the summation over j and k marginalises over all possible alignments to output position i.

Distinguishing copies The independence assumptions in Eq. (4) imply that $P(\mathbf{y}_i|\mathbf{x}_j)$ will have the same distribution for *all* copies of x_j, i.e. the model cannot express a preference to translate the first copy of *Seattle* to `city` and the second copy to `seattle` in Fig. 1. To enable this, we distinguish different copies of an input token.

We do this by defining F not as a matrix but as a tensor $F \in \mathbb{B}^{n \times l \times d}$ where d is some fixed maximum fertility value. Let $F_{i,j,u} = 1$ iff intermediate token j is the u-th copy of input token i. For example, in Fig. 1, $F_{4,6,1} = 1$ and $F_{4,7,2} = 1$. With this definition of F (and accordingly defined $\tilde{F}$), we can define a stronger decoder:

$$P(\mathbf{y}|\mathbf{x}, \tilde{F}, \tilde{R}) = \prod_{i=1}^{l} \sum_{j,k,u} P(\mathbf{y}_i|\mathbf{x}_j, u)\tilde{F}_{j,k,u}\tilde{R}_{k,i}$$

where we now additionally marginalise over which copy of the input sequence we are translating.

5 Fertility and Alignment

In this section we describe how to compute $\tilde{F} = \mathbb{E}_{P(F|\mathbf{x},l)}F$, i.e. the expected alignment that results from the fertility step given that the intermediate token sequence will have length l.

In the previous section, we looked at the fertility step mostly from the perspective of an alignment between the input tokens and the intermediate tokens. However, the fertility step is *parameterised* as assigning a fertility value $f_i \in 0, \ldots, d \in \mathbb{N}$ to every token x_i. We now show how to compute $\tilde{F}$ efficiently as a function of the distribution over fertility values $P(\mathbf{f}|\mathbf{x})$.

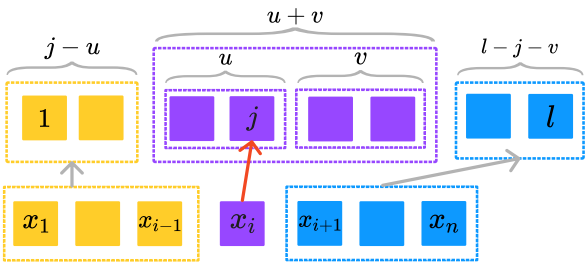

Figure 2: Efficiently computing the marginal probability that the u-th copy of i is at position j. We partition the intermediate sequence into four parts, and marginalise over all possible ways of choosing v.

We denote the alignment that follows from $\mathbf{f}$ as $F(\mathbf{f})$, i.e. $F(\mathbf{f})_{i,j,u} = 1$ iff intermediate token j is the u-th copy of token i. $\tilde{F}$ can be expressed as:

$$\tilde{F} = \mathbb{E}_{P(F(\mathbf{f})|\mathbf{x},l)}F(\mathbf{f}) \quad (5)$$

where we assume that the fertility values are independent of each other conditioned on $\mathbf{x}$:

$$P(F(\mathbf{f})|\mathbf{x}) \triangleq P(\mathbf{f}|\mathbf{x}) \triangleq \prod_{i=1}^{n} P(f_i|\mathbf{x})$$

Note that conditioning on the output length l in Eq. (5) introduces inter-dependencies between the values. In order to compute $\tilde{F}_{i,j,u}$, we need to marginalise over all possible assignments to the fertility vector $\mathbf{f}$ which satisfy $F(\mathbf{f})_{i,j,u} = 1$. We do this with a dynamic programming algorithm that is similar to the forward/backward algorithm for HMMs (Baum, 1972).

Computing marginals We characterise the situations where $F(\mathbf{f})_{i,j,u} = 1$ as the integer solutions to a set of equations (see also Fig. 2). First, in order for intermediate token j to be the u-th copy of i, there should be $j-u$ intermediate tokens generated by input tokens preceding i:

$$f_1 + \ldots + f_{i-1} = j - u \quad (6)$$

Second, the fertility f_i has to be at least u. We capture this by requiring

$$f_i = u + v \quad (7)$$

with $v \geq 0$. Finally, the input tokens following i have to contribute the remaining $l - j - v$ intermediate tokens:

$$f_{i+1} + \ldots + f_n = l - j - v \quad (8)$$

We then compute the probability of creating a sequence of length l, where the j-th intermediate

token is the u-th copy of i, by marginalising over v (dropping $\mathbf{x}$ for readability):

$$P(F(\mathbf{f})_{i,j,u} = 1, f_0 + \ldots + f_{n+1} = l) \quad (9)$$
$$= \sum_{v=0}^{\min(l-j,d-u)} P(f_0 + \ldots + f_{i-1} = j - u) \times$$
$$P(f_i = u + v) P(f_{i+1} + \ldots + f_{n+1} = l - j - v)$$

We handle the boundary cases with $i = 1, i = n$ by adding dummy variables f_0 and f_{n+1} and setting $P(f_0 = 0) = P(f_{n+1} = 0) = 1$.

In order to compute Eq. (9), we also compute 'forward' probabilities $P(f_0 + \ldots + f_i = h)$ and 'backward' probabilities $P(f_i + \ldots + f_{n+1} = h)$ for all i, h. This can be done recursively:

$$P(f_0 + \ldots + f_i = h) =$$
$$\sum_{r=0}^{d} P(f_i = r) P(f_0 + \ldots + f_{i-1} = h - r)$$

Finally, $\tilde{F}_{i,j,u} = 1/P(f_0 + \ldots + f_{n+1} = l) \times P(F(\mathbf{f})_{i,j,u} = 1, f_0 + \ldots + f_{n+1} = l)$ because $\tilde{F}_{i,j,u}$ was defined in Eq. (5) by conditioning on the length l. We provide pseudo-code for all steps in Appendix A.

Runtime The runtime of the fertility step is dominated by computing Eq. (9). Note that v has a range of at most d, thus, we can compute the probabilities for all i, j, u in $O(n \cdot l \cdot d^2)$ time. Eq. (9) can be computed in parallel for each i, j, u.

6 Composing Fertility and Reordering

We will now describe in detail how $P(F|\mathbf{x})$ and $P(R|\mathbf{x}, \tilde{F})$ are defined and how the fertility and reordering layers are composed.

6.1 Fertility Layer

The fertility layer takes a desired output length l and a sequence of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as input (GloVe embeddings (Pennington et al., 2014) in our case) and returns a marginal alignment $\tilde{F}$ (see Section 5) and a sequence of vectors $\mathbf{h}_1, \ldots, \mathbf{h}_l$ for use as input to the next layer. We compute the distribution over fertilities by first encoding $\mathbf{x}_1, \ldots, \mathbf{x_n}$ with a bidirectional LSTM, yielding a sequence of hidden states $\mathbf{h}^f_1, \ldots, \mathbf{h}^f_n$. We then model $P(f_i|\mathbf{x}) = \text{softmax}_\tau(\text{MLP}^f(\mathbf{h}^f_i))$, where τ is the temperature parameter of the softmax.

We use $\tilde{F}$ (Eq. (5)) as a form of structured attention to compute the input to the reordering layer:

$$\mathbf{h}_j = \sum_{i,u} \tilde{F}_{i,j,u}(\mathbf{x}_i + \mathbf{w}_u)$$

where $\mathbf{w}_u$ is a learned embedding indicating that j is a u-th copy of some token. Intuitively, $\mathbf{h}_j$ for an intermediate token j represents the corresponding token in the input sequence and also indicates *which* copy of that token it is.

6.2 Reordering Layer

Given the output $\mathbf{h}_1, \ldots, \mathbf{h}_l$ from the fertility layer, the reordering layer computes the alignment distribution $\tilde{R}$ as the expected permutation following Wang et al. (2021). This procedure involves populating a CYK-style chart with scores. We first run a bidirectional LSTM with a skip connection over $\mathbf{h}$ and compute a contextualised representation of the tokens after the fertility step:

$$\mathbf{h}^r_i = [\text{LSTM}^r(\mathbf{h}_{\leq i}), \text{LSTM}^r(\mathbf{h}_{\geq i})] + \mathbf{h}_i$$

Based on these representations, we compute scores for the chart following Stern et al. (2017).

6.3 Decoder

Our decoder factors as follows (see Section 4):

$$P(\mathbf{y}|\mathbf{x}, \tilde{R}, \tilde{F}) = \prod_{i=1}^{l} \underbrace{\sum_{j,k,u} P(\mathbf{y}_i|\mathbf{x}_j, u) \tilde{F}_{j,k,u} \tilde{R}_{k,i}}_{P(\mathbf{y}_i|\mathbf{x}, \tilde{R}, \tilde{F})}$$

$P(\mathbf{y}_i|\mathbf{x}_j, u)$ conditions only on the original input token and on the index indicating which copy of this token we are translating. For this reason, we contextualise the *input* with a bidirectional LSTM with a skip connection:

$$\mathbf{h}'_j = \rho[\text{LSTM}^d(\mathbf{x}_{\leq j}), \text{LSTM}^d(\mathbf{x}_{\geq j})] + \mathbf{x}_j \quad (10)$$

with ρ as hyperparameter.

We experiment with three versions of the decoder. In (i), we parameterise $P(\mathbf{y}_i|\mathbf{x}_j, u)$ as $P(\mathbf{y}_i|\mathbf{x}_j, u) = \text{softmax}(W_u \text{MLP}(\mathbf{h}'_j))$. In (ii), we additionally use a copy mechanism (Gu et al., 2016). In (iii), we use an autoregressive variant where we encode $\mathbf{y}_{<i}$ with an LSTM, defining $P(\mathbf{y}_i|\mathbf{y}_{<i}, u) = \text{softmax}(W_u \text{MLP}(\mathbf{h}'_j + \text{LSTM}(\mathbf{y}_{<i}))$.

6.4 Training

As mentioned in Section 4, at training time we condition on the observed length l of $\mathbf{y}$: $P(\mathbf{y}|\mathbf{x}) = P(l|\mathbf{x})P(\mathbf{y}|\mathbf{x}, \tilde{F}, \tilde{R})$ with $\tilde{F}$ conditioned on l. We use a weighted version of the log likelihood as the objective function:

$$\sum_i \lambda_1 \log P(l^i|\mathbf{x}^i) + \log P(\mathbf{y}^i|\mathbf{x}^i, l^i)$$

with i ranging over the training examples.

For the semantic parsing tasks, we found it necessary to give our model a reasonable starting point in terms of alignments. We encourage it to respect high-confidence automatic alignments during the first m training epochs by adding the following term to our objective function:

$$\lambda_2 \sum_{(i,j)\in\mathcal{A}} \log \sum_{k,u} \tilde{F}_{i,k,u}\tilde{R}_{k,j}$$

where $\mathcal{A}$ is the set of alignments with a posterior probability of at least χ according to an IBM-1 alignment model (Brown et al., 1993).

Initialising an alignment model with alignments from a simpler model was a common strategy in statistical machine translation (Och and Ney, 2003).

6.5 Inference

In order to make predictions with a trained model, we want to compute the most likely output $\mathbf{y}$ given the input $\mathbf{x}$. It is convenient to treat the length as a discrete variable and use the same algorithm for computing $\tilde{F}$ as derived for training. We therefore search for $\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_l P(l|\mathbf{x})P(\mathbf{y}^l|\mathbf{x}, l)$ with $\mathbf{y}^l = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}, \tilde{F}, \tilde{R})$. For any given l we can easily find $\mathbf{y}^l$ in versions (i) and (ii) of the decoder:

$$\mathbf{y}^l_i = \arg\max_{\mathbf{y}_i} P(\mathbf{y}_i|\mathbf{x}, \tilde{R}, \tilde{F})$$

For version (iii) of the decoder, we use greedy search instead. It would be too costly to compute $\mathbf{y}^l$ for all l, so we explore only the top k most likely lengths.

Grammar-based decoding In executable semantic parsing, we want to produce only well-formed outputs, which can be characterised by a context-free grammar G. In practical applications, this grammar is needed to execute the query, so there is little extra engineering effort in using it for decoding. We search for

$$\mathbf{y}^l = \arg\max_{\mathbf{y}\in L(G)} P(\mathbf{y}|\mathbf{x}, l)$$

Because of the simple decoder we can do this exactly by applying a modified version of Viterbi CYK. Unlike in parsing though, the string is not observed because it is exactly what we are looking for. Therefore, we fill all entries from i to i in the chart C with $C_{A,i,i} = \max_{A\to a\in G} P(y_i = a|\mathbf{x}, \tilde{R}, \tilde{F})$. We then continue with the normal Viterbi CYK with weights of 1 on all other rules.

7 Evaluation

7.1 Synthetic Data

In order to probe the expressive capacity and inductive bias of our model, we evaluate on mirroring task $T = \{(w, ww^R)|w \in \Sigma^*\}$, e.g. $abc \to abccba$. The challenge is that the length of the dependency between output tokens grows with the length of the example. Models are trained on examples with input lengths 3 to 9, and tested in two setups. In the first setup (Length), the model has to generalise to examples with lengths 11 to 20; we use examples with length 10 as validation data. In the second setup (unseen combination, UC), the model only sees the symbols `x`, `y` and `z` grouped together as `xyz` on the input side at training time. The model is tested on examples that contain `x`, `y` or `z` adjacent to other symbols. See Appendix B.1 for further details on the setup.

We compare our model without copying (F→R) with a variant that first applies the reordering and then the fertility step (R→F) and autoregressive variants of the two (AR F→R and AR R→F). As baselines, we also compare with an LSTM-based seq2seq model with attention and a Transformer with relative positional embeddings, which was previously shown by Csordás et al. (2021) to perform well at compositional generalisation. AR R→F has similarities to Wang et al. (2021) who first reorder and then use an autoregressive decoder with monotonic attention.

Results Table 1 shows mean accuracy across 5 random initialisations. The accuracy of the relative Transformer and the LSTM-based seq2seq model drops sharply for longer inputs. In contrast, F→R and AR F→R generalise perfectly even to much longer examples. In the UC setup, F→R outperforms the rest by a wide margin. Interestingly, all

Model	Accuracy					
		Length				UC
	dev	11	12	13	14 - 20	test
Transformer	82.0	3.0	0.0	0.0	0.0	42.0
LSTM	**100.0**	94.3	67.9	6.9	0.0	0.1
R→F	0.0	0.0	0.0	0.0	0.0	1.3
AR R→F	90.0	85.7	89.8	87.5	80.5	1.2
F→R	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**79.9**
AR F→R	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	32.1

Table 1: Exact match accuracy on the mirroring task.

autoregressive models struggle in this setup, including AR F→R which obtained perfect accuracy in the Length setup. This is consistent with the hypothesis that autoregressive models tend to memorise entire chunks (Hupkes et al., 2020).

Expressivity of F→R and R→F For this task, an input token corresponds to two output tokens that may be arbitrarily far apart from each other. F→R can learn this alignment because this can be captured by a separable permutation (see Section 3.2) following the fertility step duplicating every input token. In contrast, R→F and AR R→F cannot represent the correct alignment directly because the fertility step is applied only after the reordering, leading to alignments between an input token and a *contiguous span* in the output. However, in theory, they can represent this alignment *implicitly* through the LSTM (Eq. (10)). The evaluation shows that this does not work reliably in practice: we find that R→F gets stuck in bad local minima and fails completely on the task. While AR R→F performs well in the Length setup, it is weak in the UC setup.

7.2 Geoquery

Geoquery (Zelle and Mooney, 1996) is a standard dataset for semantic parsing and has recently been used to evaluate to what extent semantic parsers are capable of generalising to (i) structurally unseen queries (template split), and (ii) structurally unseen long examples. We follow the setup of Herzig and Berant (2021), using the variable-free FunQL representation (Kate et al., 2005), a copy mechanism, and evaluate with execution accuracy.

Results Table 2 shows the results on the different splits. We report means and standard deviations of 5 random initialisations. Our method performs well across the different splits, and in particular on the length split that evaluates a challenging form of compositional generalisation. As an ablation, we remove the grammar-based decoding.

Model	iid	Template	Length
Seq2Seq ‡ (HB)	78.5	46.0	24.3
Seq2Derivation (HB)	72.1	54.0	24.6
BART-base‡ (HB)	87.1	67.0	19.3
Span (HB)	78.9	65.9	41.4
Span + lexicon (HB)	86.1	82.2	63.6
Liu et al. (2021)‡	-	**84.1**	-
Wang et al. (2021)‡	75.2*	43.2*	-
R→F	**89.1**±1.0	80.4±1.2	68.6±1.4
R→F ‡	83.5±0.7	73.0±1.6	49.2±5.1
F→R	88.6±3.3	79.9±2.5	**68.8**±5.2
F→R ‡	80.7±2.3	68.7±4.3	53.4±5.9
AR F→R ‡	81.1±1.0	52.8±3.3	37.6±1.8

Table 2: Accuracy on different splits of Geoquery. HB is Herzig and Berant (2021) and ‡ refers to systems that do not enforce well-formedness of the output. * Wang et al. (2021) use exact match accuracy and anonymise named entities instead of copying.

We notice a considerable drop in accuracy but it still outperforms the baselines. The drop in accuracy is slightly bigger out-of-distribution than in-distribution.

In line with the experiments on synthetic data, AR F→R and the approach of Wang et al. (2021) drastically lose accuracy when going from in-distribution to the compositional generalisation setups. This provides further evidence that a strong decoder can hinder compositional generalisation.

In contrast to the experiments on synthetic data, R→F and F→R perform comparably. Manual inspection of the data shows that good alignments on this dataset can be obtained even with the stronger assumption on possible alignments made by R→F.

7.3 ATIS

ATIS (Dahl et al., 1994) is a semantic parsing datasets for flight bookings. In comparison to Geoquery, the queries tend to be longer and the word order is more flexible. We use the variable-free FunQL notation as annotated by Guo et al. (2020). Apart from the original iid test split, we create a length split: Semantic parses with fewer than 4 conjuncts form the training set, parses with exactly 4 conjuncts form the development set and the test set contains instances with more than 4 conjuncts. Details on the split and preprocessing are in Appendix B.3.

We compare our model with finetuned BART-base (Lewis et al., 2020), an LSTM-based seq2seq model with attention and the relative Transformer

Model	iid	Length
LSTM seq2seq	76.52±1.66	4.95±2.16
LSTM seq2seq‡	75.98±1.30	4.95±2.16
Rel. Transformer‡	75.76±1.43	1.15±1.41
BART-base‡	**86.96**±1.26	19.03±4.57
F→R	74.15±1.35	**35.41**±4.09
F→R ‡	68.26±1.53	29.91±2.91

Table 3: Accuracy on different splits of ATIS.

Model	Calendar	Document	Email
BART-base‡	36.7±3.0	2.7±2.1	20.5±9.8
F→R	**69.5**±13.9	**42.4**±5.7	**55.6**±2.7
F→R ‡	57.2±19.9	36.1±5.6	43.9±3.8

Table 4: Accuracy on length splits by domain on Okapi.

(Csordás et al., 2021). We also run a version of the LSTM-based model with a large beam of size 50 and filter our instances that are not well-formed; the resulting outputs are well-formed at least 99.7% of the time.

Results Table 3 shows mean accuracy and standard deviations of 5 random initialisations. While on the iid split, our approach does not quite reach the same accuracy as the baselines, it outperforms them on the compositional length split by a margin of more than 16 points. Without grammar-based decoding, we again observe a noticeable loss in accuracy but we still substantially outperform BART on the length split. Constraining the output to be grammatical does not appear as beneficial for the LSTM baseline.

7.4 Okapi

Hosseini et al. (2021) introduce a semantic parsing dataset for evaluating compositional generalisation on three domains (document, calendar, email). Since template splits were not found to be challenging, we focus on generalising to longer examples. Models are trained on short examples with up to 3 'parameters' (such as filtering based on an attribute or ordering results) and are tested on examples with more parameters (at least 4 for the calendar and email domain and at least 5 for the document domain). The splits are described in Appendix B.4. In contrast to the other datasets we consider, Okapi is noisy because it was collected with crowd workers. This presents an additional challenge.

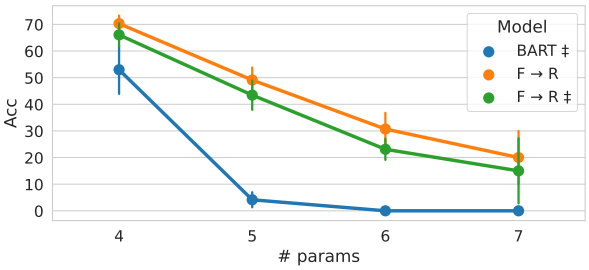

Figure 3: Accuracy on the document domain of Okapi by number of parameters in the gold logical form.

Results Table 4 shows the results of our model with copying from 5 random initialisations. F→R outperforms fine-tuned BART-base by a large margin, in particular on the challenging split of the document domain.

Fig. 3 shows the accuracy on the development and test set of the document domain as a function of the number of parameters in the gold logical form. BART performs relatively well when applied to examples with one more parameter than seen in the training set but then its performance drops sharply. F→R is more accurate and its accuracy also drops much slower with the number of parameters. We notice different failure modes for the two models: on the test set, BART deviates by 5.4 tokens on average from the gold length, whereas F→R deviates only by 1.0 tokens on average. This is in line with the observations of Newman et al. (2020) that seq2seq models systematically predict the end of sequence token too early on long out-of-distribution examples. Our results suggest that predicting length as a sum of fertilities is more robust towards this shift in distribution.

7.5 Style Transfer

In addition to being important for compositional generalisation, structural inductive biases can help when only little data is available. We evaluate our model in such a scenario on the style transfer tasks of Lyu et al. (2021). A model is given an English sentence and asked to reformulate it to conform with a certain 'style'. We focus on the tasks identified as challenging by Lyu et al. (2021): *active to passive* (2462 training examples), *adjective emphasis* (627 examples) and *verb emphasis* (1081 examples).

For the emphasis tasks, the word to be emphasised is provided in the input. Following Kim (2021), to incorporate it, we add a special learned embedding vector to the embedding of that token.

Transfer Type	Approach	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
Active to passive	GPT-2 (Lyu et al., 2021)	47.6	32.9	23.8	18.9	21.6	46.4	1.820
	Seq2seq (Kim, 2021)	83.8	73.5	67.3	59.8	46.7	77.1	5.941
	Neural QCFG (Kim, 2021)	83.6	77.1	71.3	66.2	49.9	80.3	6.410
	Human (Lyu et al., 2021)	93.1	88.1	83.5	79.5	58.7	90.5	8.603
	F→R	**92.1**±1.3	**85.4**±1.5	**79.7**±1.7	**74.6**±1.9	**55.9**±0.5	**86.0**±1.1	**7.610**±0.130
Adj. emphasis	GPT-2 (Lyu et al., 2021)	26.3	7.9	2.8	0.0	11.2	18.8	0.386
	Seq2seq (Kim, 2021)	50.5	29.6	18.4	11.9	24.2	51.4	1.839
	Neural QCFG (Kim, 2021)	67.6	50.6	39.3	31.6	37.3	68.3	3.424
	Human	83.4	75.3	67.9	66.1	52.2	81.1	6.796
	F→R	**78.3**±1.4	**61.9**±0.7	**49.9**±1.3	**40.5**±1.4	**43.0**±1.0	**69.1**±0.6	**4.268**±0.170
Verb emphasis	GPT-2 (Lyu et al., 2021)	30.9	17.0	9.5	4.1	14.0	29.2	0.593
	Seq2seq (Kim, 2021)	52.6	38.9	29.4	21.4	29.4	46.4	2.346
	Neural QCFG (Kim, 2021)	66.4	51.2	40.7	31.9	37.0	**58.9**	3.227
	Human (Lyu et al., 2021)	64.9	56.9	49.3	42.1	43.3	69.3	5.668
	F→R	**68.4**±0.6	**52.7**±0.6	**41.6**±0.7	**32.8**±0.6	**37.4**±0.4	**58.9**±0.4	**3.498**±0.121

Table 5: Results on the hard style transfer tasks from Lyu et al. (2021). All models except for GPT2 use copying.

Results Table 5 shows the results comparing our F→R model to previous work based on three random initialisations. We achieve state-of-the-art results on all style transfer tasks on all metrics. The improvement compared to prior work is strongest for the active-to-passive task and weakest for the verb emphasis task, where our model ties with Kim (2021) in terms of ROUGE-L.

8 Conclusion

We presented a flexible end-to-end differentiable model for structured NLP tasks. It predicts the output sequence from the input by composing a fertility layer with a reordering layer. The evaluation shows that our model performs well in structural generalisation setups, in particular when the model has to generalise to longer examples than seen during training. In contrast, the accuracy of standard seq2seq models drops sharply on longer examples.

The efficient fertility layer introduced in this work may be useful in other scenarios as well, e.g. in non-autoregressive machine translation, or for (unsupervised) sentence compression when the fertility is restricted to 0 or 1. Future work could also investigate other structured layers and the best ways of composing and training them.

9 Limitations

The fertility layer is efficient but a limitation of the model presented in this paper is the high runtime complexity of the reordering layer. It makes it impractical for long output sequences (e.g. more than 50 tokens). While the structured reordering step can represent many permutations of practical interest, we observed a small number of cases where our model could not produce the correct permutation. We note that our approach is modular and the reordering layer could be replaced by a faster one with fewer restrictions in future work, e.g. based on one-to-one matchings.

While our method obtains strong accuracy in compositional generalisation setups without contextualised encoders, it remains an open question how different ways of integrating contextualised encoders affect the performance of our method in compositional generalisation setups. In addition, it is an open question how a *pretrained decoder* would influence our model's ability to generalise compositionally since we found that a non-pretrained LSTM-based decoder can be detrimental.

Acknowledgements

We thank Bailin Wang and Jonas Groschwitz for technical discussions and we thank Agostina Calabrese and Verna Dankers for comments on this paper.

ML is supported by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1), the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences, and a grant from Huawei Technologies. ML and IT acknowledge the support of the European Research Council (ERC StG BroadSem 678254). IT is also supported by the Dutch National Science Foundation (NWO Vici VI.C.212.053).

References

Ekin Akyürek and Jacob Andreas. 2021. Lexicon learning for few shot sequence modeling. In *Proceedings of the ACL-IJCNLP (Volume 1: Long Papers)*, pages 4934–4946, Online.

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

Leonard E Baum. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8.

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

Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. 1998. Pattern matching for permutations. *Information Processing Letters*, 65(5):277–283.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 876–885, San Diego, California.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. In *Proceedings of the ACL-IJCNLP (Volume 1: Long Papers)*, pages 3322–3335, Online.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Proceedings of the 2021 EMNLP*, pages 619–634, Online and Punta Cana, Dominican Republic.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany.

Jiaqi Guo, Qian Liu, Jian-Guang Lou, Zhenwen Li, Xueqing Liu, Tao Xie, and Ting Liu. 2020. Benchmarking meaning representations in neural semantic parsing. In *Proceedings of the 2020 EMNLP (EMNLP)*, pages 1520–1540, Online.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Proceedings of the ACL-IJCNLP (Volume 1: Long Papers)*, pages 908–921, Online.

Saghar Hosseini, Ahmed Hassan Awadallah, and Yu Su. 2021. Compositional generalization for natural language interfaces to web apis. *arXiv preprint arXiv:2112.05209*.

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

Rohit J. Kate, Yuk Wah, Wong Raymond, and J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of AAAI-05*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 EMNLP (EMNLP)*, pages 9087–9105, Online.

Yoon Kim. 2021. Sequence-to-sequence learning with latent neural grammars. In *Advances in Neural Information Processing Systems*, volume 34, pages 26302–26317. Curran Associates, Inc.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training

for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the ACL*, pages 7871–7880, Online.

Philip M Lewis and Richard Edwin Stearns. 1968. Syntax-directed transduction. *Journal of the ACM (JACM)*, 15(3):465–488.

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

Yiwei Lyu, Paul Pu Liang, Hai Pham, Eduard Hovy, Barnabás Póczos, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2021. StylePTB: A compositional benchmark for fine-grained controllable text style transfer. In *Proceedings of the 2021 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 2116–2138, Online.

Chaitanya Malaviya, Pedro Ferreira, and André F. T. Martins. 2018. Sparse and constrained attention for neural machine translation. In *Proceedings of the 56th Annual Meeting of the ACL (Volume 2: Short Papers)*, pages 370–376, Melbourne, Australia.

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D. Manning. 2020. The EOS decision and length extrapolation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 276–291, Online.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In *Findings of the ACL: EMNLP 2020*, pages 2482–2495, Online.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 EMNLP (EMNLP)*, pages 1532–1543, Doha, Qatar.

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

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *International Conference on Machine Learning*, pages 2837–2846. PMLR.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the ACL-IJCNLP (Volume 1: Long Papers)*, pages 922–938, Online.

David Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 23–30, New York City.

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

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Structured reordering for modeling latent alignments in sequence transduction. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Annual Meeting of the ACL*, pages 1530–1537, Florence, Italy.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 EMNLP*, pages 1307–1316, Austin, Texas.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

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

Algorithm 1 Forward

Require: f $\in \mathbb{R}^{n \times d}$, normalised over the second axis, output sequence length l

```
function FWD(f, l)
    Init fwd ∈ R^{n×l} with zeros
    for h in range(0, d) do                    ▷ Base case
        fwd[0, h] = f[0, h]
    for i in range(1, n) do                    ▷ Recursion
        for h in range(0, l) do
            for r in range(0, min(d, h+1)) do
                fwd[i, h] += fwd[i-1, h-r] × f[i, r]
    return fwd
```

Algorithm 2 Marginals

Require: f $\in \mathbb{R}^{n \times d}$, normalised over the second axis, output sequence length l

```
function MARGINALS(f, l)
    Init F ∈ R^{n×l×d} with zeros
    fwd = fwd(f, l)
    bwd = rev(fwd(rev(f), l))
    for i in range(0, n-1) do
        for j in range(0, l) do
            for u in range(1, min(d, j+1+1)) do
                for v in range(0, min(d-u, l-j)) do
                    F[i, j, u] += fwd[i, j+1-r1] ×
                        f[i,u+v]×bwd[i+1,l-(j+1+v)]
    divide all entries in F by fwd[i, l]
    return F
```

A Pseudo code

We present the algorithms for computing $\tilde{F}$ in python-style pseudo code, using 0-based indexing and `range(i,k)` refers to the integers $i, \ldots, k-1$. As in the main paper, d refers to the maximum fertility value (hyperparameter).

Algorithm 1 shows how to compute all forward probabilities. If `rev(f)` reverses matrix `f` on the first dimension (e.g. like `torch.flip`), then we can compute the backward probabilities as `rev(fwd(rev(f), l))`. Algorithm 2 shows how $\tilde{F}$ can be computed.

B Data, grammars, pre- and post-processing

B.1 Synthetic data

In both setups, we generate 4000 training examples, 200 development examples and 1000 test examples. We use an alphabet size of $|\Sigma| = 11$ for the Length setup and $|\Sigma| = 11 + 3$ for the UC setup to accommodate for `x,y,z`. Symbols are chosen uniformly at random. In the Length setup, we choose the length of the example uniformly at random. In the UC setup, we do so as well but with probability 0.2 we insert an `xyz` cluster if this does not exceed the maximum length.

In the UC setup, we use development data from the training distribution.

We do not use grammar-based decoding or copying on the synthetic data.

B.2 Geoquery

We remove all parentheses in the logical forms, as they can be restored in post-processing. We also remove quotes around named entities in preprocessing to enable copying (`'texas'` becomes `texas`) and restore them in post-processing. Following Herzig and Berant (2021), we only allow copying of named entities and do not copy predicate symbols (e.g. `river`).

We use the grammar of Wong and Mooney (2006) for decoding as provided by Guo et al. (2020).

B.3 ATIS

We found that a naive length split led to having very few examples in the training set that used a date since both the month and the day count as one conjunct each. Therefore, we created 33 templates with three conjuncts based on existing ATIS examples (with four or more conjuncts) that contain dates and add 8 instances of each template with randomly chosen dates to the training set. In addition, we removed any exact duplicate samples from the data.

Similarly to Geoquery, we remove those parentheses that can be deterministically recovered in post-processing. However, in contrast to Geoquery the parentheses for `or` and `intersection` need to be kept because the arities of those operators are not fixed. We run all our experiments on this representation of ATIS.

For grammar-based decoding we use the "typed" grammar provided by Guo et al. (2020) and do not use the copy mechanism.

B.4 Okapi

We found there was too much distributional overlap in the original length split provided by Hosseini et al. (2021) and therefore use our own split:

For the document domain, our development set contains examples with 4 parameters, and the test

Dataset	Split/Version	Train	Dev	Test
Geoquery	iid	540	60	280
	template	544	60	276
	length	540	60	280
ATIS	iid	4465	497	448
	length	4017	942	331
Okapi/length	Calendar	1145	200	1061
	Document	2328	412	514
	Email	2343	200	991
Style transfer	active to passive	2462	136	137
	adjective emphasis	627	34	35
	verb emphasis	1081	60	60

Table 6: Number of examples per dataset/split.

set contains examples with at least 5 parameters. For the other two domains, there is insufficient data for such out-of-distribution development sets. Therefore, we chose our test set to contain all examples with at least 4 parameters and our development sets to consist of 95% in-distribution data and 5% of examples from the the examples with 4 parameters (which makes the bulk of the test distribution).

We manually create a grammar of well-formed logical forms for the three domains of Okapi (included in the code).

B.5 StylePTB

For comparability, we also tokenise on whitespace following Kim (2021). We do not restrict the output of the model with a grammar.

C Details on evaluation metrics

We provide code for all evaluation metrics in our repository/dependencies.

Geoquery We use the code of Herzig and Berant (2021) to compute execution accuracy (`https://github.com/jonathanherzig/span-based-sp`).

ATIS We allow for different order of conjuncts between system output and gold parse in computing accuracy. We do this by sorting conjuncts before comparing two trees node by node.

Okapi We follow Hosseini et al. (2021) and disregard the order of the parameters for computing accuracy. Since Hosseini et al. (2021) did not make their evaluation code publicly available, we use our own implementation. Our implementation uses sets and does not punish a model for predicting a correct parameter multiple times. For example, if the gold logical form contains `FILTER message.isRead eq False`, a necessary condition for a prediction to count as correct is that it must contain this string *at least* once.

StylePTB We follow Kim (2021) and use `https://github.com/Maluuba/nlg-eval` (commit `7f79930`) for all evaluation metrics, ensuring that BLEU, ROUGE and METEOR are scaled to 0 - 100.

D Hyperparameters

We provide a configuration file for each of our models with the chosen hyperparameters in our code repository (`configs/`). We set the maximum fertility value d to $d = 4$ for all datasets except for the style transfer tasks where we set it to $d = 3$.

At test time, we explore the top $k = 5$ most likely lengths when using grammar-based decoding. Without grammar-based decoding we used $k = 1$ as using $k = 5$ provided little improvement.

Many but not all instances of Geoquery require identity permutations. We found this to lead to the issue that the model gets stuck in a very steep local minimum within the first epoch where it would predict only identity permutations. We fixed this issue by reducing the learning rate in the feedforward network that predicts the scores for the permutation trees to 1×10^{-6}.

D.1 Hyperparameter selection

We select hyperparameters using a combination of manual selection and a random search. We optimise hyperparameters for accuracy on the development set of compositional generalisation splits, where available, (Length setup for the synthetic data, template split for Geoquery), and then use those hyperparameters for all splits of a (domain of a) dataset.

The high variance in accuracy across random initialisations often observed in compositional generalisation setups makes it difficult to tune hyperparameters even if an out-of-distribution development set exists. We restrict random hyperparameter search to two random seeds. After the hyperparameter search, we pick the two most promising configurations (according to (execution) accuracy), pick a new random seed and train them again to choose the one which provides the most stable accuracy (approximated as the highest accuracy on the new seed). We then run the main experiments

Dataset	Model	# params
Mirror	F→R and R→F	1.019
	AR F→R and AR R→F	1.12
	LSTM	3.187
	Transformer	33.067
Geoquery	F→R and R→F	2.541
	AR F→R	2.765
ATIS	LSTM	4.669
	Transformer	58.468
	F→R	3.511
Okapi/Calendar	F→R	2.466
Okapi/Document	F→R	2.468
Okapi/Email	F→R	2.485
Style/Active to passive	F→R	2.814
Style/Adjective emphasis	F→R	2.698
Style/Verb emphasis	F→R	3.133

Table 7: Number of parameters in millions in our models.

with the chosen hyperparameters and completely new random seeds.

We randomly sample 20 configurations per hyperparameter search. Since this procedure is expensive, we do not run train our models fully to convergence. The bounds of the hyperparameter search are reported in our repository.

For all our models, we initially chose hyperparameters manually, and then ran a random hyperparameter search as described above. If the manually chosen hyperparameters resulted in same or better performance (on the development set) on average, we kept those, and otherwise used the ones found by the hyperparameter search.

For example, on Geoquery, we noticed a particular sensitivity to hyperparameters, and the manually selected hyperparameters for F→R performed best with low variance, whereas for R→F, the hyperparameters found by the random search were better than the manually chosen ones. We think this sensitivity is at least in part caused by the small size of the dataset.

Style transfer In contrast to the other tasks we evaluate on, we did not run a hyperparameter search for the style transfer tasks and use the same manually determined hyperparameters for F→R across all style transfer tasks.

E Computing infrastructure and runtime

All experiments were run on GeForce GTX 1080 Ti or GeForce GTX 2080 Ti with 12GB RAM and Intel Xeon Silver or Xeon E5 CPUs.

The runtime of one run contains the time for training, evaluation on the devset after each epoch and running the model on the test set. We show the runtime the models we train in Table 8.

Dataset	Model	Epochs	Runtime
Mirror	F→R	7	8 min
	R→F	20	8 min
	AR F→R	20	30 min
	AR R→F	20	20 min
	Transformer	200	15 min
	LSTM	60	4 min
Geo	F→R	100	20 min
ATIS	F→R	100	11-12h
	Transformer	20	10 min
	LSTM	20	10 min
	BART	50	1.3h
Okapi / Calendar	F→R	70	1 h
	BART	40	15 min
Okapi / Document	F→R	70	1.5 h
	BART	60	30 min
Okapi / Email	F→R	70	1.5 h
	BART	40	30 min
Active → Passive	F→R	60	1 h
Adj. emphasis	F→R	60	20 min
Verb emphasis	F→R	60	30 min

Table 8: Average total runtime of the models we train. For comparison on Geoquery, Herzig and Berant (2021) report a runtime of 2 hours on comparable hardware to ours.

F Additional results

Geoquery Table 10 shows *exact match* accuracy of our models for comparison and Table 11 shows results on the development set.

ATIS Table 13 shows results on the development set. Table 12 shows the average deviation from the gold length.

Okapi Table 14 shows results on the development set.

StylePTB Table 9 shows results on the development set.

Transfer Type	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
Active to passive	94.1±0.3	88.6±0.2	83.8±0.2	79.7±0.3	57.6±0.1	87.9±0.5	7.883±0.061
Adj. emphasis	78.4±1.2	64.3±1.0	54.5±0.7	46.9±0.9	44.0±0.4	69.5±0.6	4.809±0.113
Verb emphasis	67.6±0.5	51.2±0.7	40.6±0.9	32.7±1.3	36.8±0.3	59.1±0.7	3.587±0.118

Table 9: Development accuracy of our F→R model with copying on StylePTB. We report means and standard deviations of three random initialisations.

Model	iid	Template	Length
R→F	83.5±0.7	72.7±2.0	54.0±4.0
R→F ‡	76.4±1.4	65.8±2.2	39.4±6.2
F→R	83.4±2.6	72.0±3.3	55.2±5.3
F→R ‡	76.9±2.2	63.6±4.7	46.0±6.6
AR F→R ‡	77.4±1.3	47.5±4.2	34.4±2.2

Table 10: Exact match accuracy on the splits of **Geoquery**. We report mean and standard deviation of the 5 random initialisations shown in the main paper.

Model	iid	Template	Length
F→R ‡	84.3±3.0	85.3±1.4	83.7±1.4
R→F ‡	83.7±2.2	83.0±3.0	79.0±3.2
AR F→R ‡	83.7±2.7	81.7±2.0	87.3±2.5

Table 11: Mean and standard deviations of execution accuracy on the development sets of the **Geoquery** splits (without grammar-based decoding).

Model	iid	Length
LSTM seq2seq ‡	0.46±0.09	5.39±0.42
Rel. Transformer ‡	0.49±0.07	6.19±0.52
BART-base ‡	0.24±0.02	3.40±0.25
F→R ‡	0.41±0.06	1.49±0.18

Table 12: Means and standard deviations of the average absolute deviation from the gold length by model on ATIS, i.e. lower is better

Model	iid	Length
LSTM seq2seq ‡	81.53±0.77	43.86±2.93
Rel. Transformer ‡	81.53±0.44	34.84±5.28
BART-base ‡	90.54±0.45	65.29±1.01
F→R ‡	73.80±1.62	54.42±1.25

Table 13: Mean and standard deviations accuracy on the development sets of the **ATIS** splits (without grammar-based decoding).

Model	Calendar	Document	Email
BART-base ‡	94.8±0.3	56.2±10.9	91.4±0.4
F→R ‡	86.4±3.4	66.1±4.8	85.0±2.8

Table 14: Mean and standard deviations accuracy on the development sets of the **Okapi** splits (without grammar-based decoding).