Compositional Generalization and Natural Language Variation: Can a Semantic Parsing Approach Handle Both?

Peter Shaw Ming-Wei Chang Panupong Pasupat Kristina Toutanova

Google Research

{petershaw,mingweichang,ppasupat,kristout}@google.com

Abstract

Sequence-to-sequence models excel at handling natural language variation, but have been shown to struggle with out-of-distribution compositional generalization. This has motivated new specialized architectures with stronger compositional biases, but most of these approaches have *only* been evaluated on synthetically-generated datasets, which are not representative of natural language variation. In this work we ask: can we develop a semantic parsing approach that handles both natural language variation and compositional generalization? To better assess this capability, we propose new train and test splits of non-synthetic datasets. We demonstrate that strong existing semantic parsing approaches do not yet perform well across a broad set of evaluations. We also propose NQG-T5, a hybrid model that combines a high-precision grammar-based approach with a pre-trained sequence-to-sequence model. It outperforms existing approaches across several compositional generalization challenges, while also being competitive with the state-of-the-art on standard evaluations. While still far from solving this problem, our study highlights the importance of diverse evaluations and the open challenge of handling both compositional generalization and natural language variation in semantic parsing.

1 Introduction

Sequence-to-sequence (seq2seq) models and related approaches have been widely used in semantic parsing (Dong and Lapata, 2016; Jia and Liang, 2016) and excel at handling the natural language variation[1] of human-generated queries. However, evaluations on synthetic[2] tasks such as SCAN (Lake and Baroni, 2018) have shown that seq2seq models generalize poorly to out-of-distribution compositional utterances, such as "jump twice" when only "jump", "walk", and "walk twice" are seen during training. This ability to generalize to novel combinations of the elements observed during training is referred to as *compositional generalization*.

This has motivated many specialized architectures with stronger compositional biases that improve peformance on SCAN (Li et al., 2019; Russin et al., 2019; Gordon et al., 2019; Lake, 2019; Liu et al., 2020; Nye et al., 2020; Chen et al., 2020). However, most of these approaches have only been evaluated on synthetic datasets. While synthetic datasets can be useful for precise and interpretable evaluation of specific phenomena, they are typically less representative of the natural language variation that a real-world semantic parsing system must handle.

In this paper, we ask: *can we develop a semantic parsing approach that handles both natural language variation and compositional generalization?* Surprisingly, this question is understudied. As visualized in Figure 1, most prior work evaluates *either* out-of-distribution compositional generalization on synthetic datasets, *or* in-distribution performance on non-synthetic datasets. Notably, designing approaches that can handle both compositional generalization and the natural language variation of non-synthetic datasets is difficult, as different approaches have their own strengths and limitations. For example, large pre-trained seq2seq models that perform well on in-distribution evaluations do not address most of the

[1]We use the term *natural language variation* in a broad sense to refer to the many different ways humans can express the same meaning in natural language, including differences in word choice and syntactic constructions.

[2]We make a coarse distinction between *synthetic* datasets, where natural language utterances are generated by a program, and *non-synthetic* datasets, where natural language utterances are collected from humans.

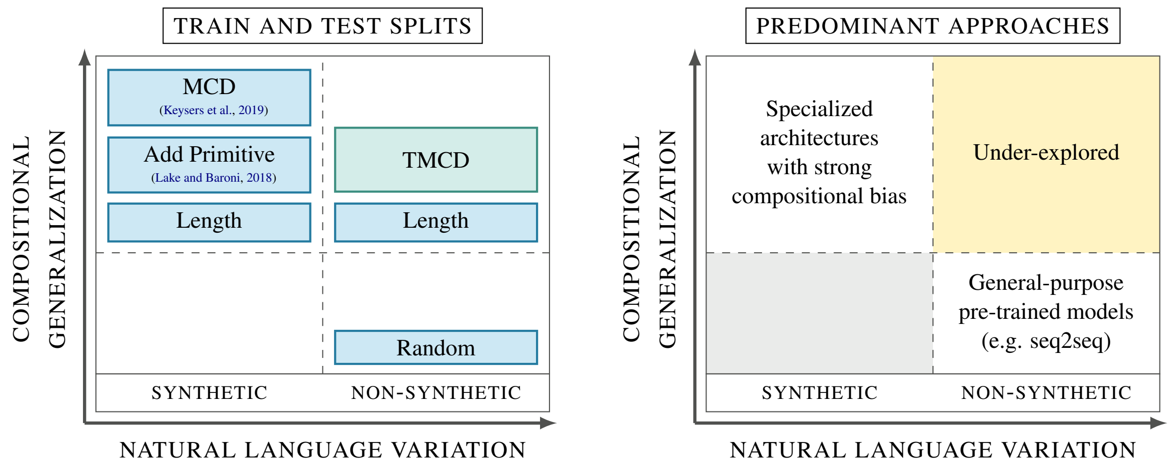
TRAIN AND TEST SPLITS
MCD
(Keysers et al., 2019)
Add Primitive
(Lake and Baroni, 2018)
Length
TMCD
Length
Random
SYNTHETIC
NON-SYNTHETIC
COMPOSITIONAL GENERALIZATION
NATURAL LANGUAGE VARIATION
PREDOMINANT APPROACHES
Specialized architectures with strong compositional bias
Under-explored
General-purpose pre-trained models (e.g. seq2seq)
SYNTHETIC
NON-SYNTHETIC
COMPOSITIONAL GENERALIZATION
NATURAL LANGUAGE VARIATION

Figure 1: **Left:** To assess both out-of-distributional compositional generalization and handling of natural language variation, we evaluate semantic parsing approaches across a diverse set of train and test splits. We introduce TMCD splits to add a challenging evaluation of compositional generalization for non-synthetic data. Ordering within each cell is arbitrary. **Right:** Existing approaches are often only evaluated on a subset of evaluations. The goal of our proposed approach, NQG-T5, is to improve performance across a diverse set of evaluations that capture both compositional generalization and natural language variation.

compositional generalization challenges proposed in SCAN (Furrer et al., 2020).

Our research question has two important motivations. First, humans have been shown to be adept compositional learners (Lake et al., 2019). Several authors have argued that a greater focus on compositional generalization is an important path to more human-like generalization and NLU (Lake et al., 2017; Battaglia et al., 2018). Second, it is practically important to assess performance on non-synthetic data and out-of-distribution examples, as random train and test splits can overestimate real-world performance and miss important error cases (Ribeiro et al., 2020).

Our contributions are two-fold. First, on the evaluation front, we show that performance on SCAN is not well-correlated with performance on non-synthetic tasks. Moreover, strong existing approaches do not yet perform well across all of the diverse set of evaluations shown in Figure 1 (left). We also propose new Target Maximum Compound Divergence (TMCD) train and test splits, extending the methodology of Keysers et al. (2019) to create challenging evaluations of compositional generalization for non-synthetic datasets.

Second, on the modeling front, we propose NQG-T5, a hybrid model that combines a high-precision grammar-based approach with T5 (Raffel et al., 2019), leading to improvements across several compositional generalization evaluations while also being competitive on the standard splits of GeoQuery (Zelle, 1995) and Spider (Yu et al., 2018). While still far from affirmatively answering our question, our study highlights the importance of a diverse set of evaluations and the open challenge of handling both compositional generalization and natural language variation.

2 Background and Related Work

In this section, we survey recent work related to evaluating and improving compositional generalization in semantic parsing.

Evaluations To evaluate a model's ability to generalize to novel compositions, previous work has proposed several methods for generating challenging train and test splits, as well as several synthetic datasets.

A widely used synthetic dataset for assessing compositional generalization is SCAN (Lake and Baroni, 2018), which consists of natural language commands (e.g., "turn left twice") mapping to action sequences (e.g., "`I_TURN_LEFT I_TURN_LEFT`"). One of the train and test splits proposed for SCAN is the length split, where examples are separated by length such that the test set contains longer examples than the training set. Another type of split is the primitive split, where a given primitive (e.g., "jump") is seen in only a single context during training, but the test set consists of the primitive recombined with other elements observed during training (e.g., "jump twice"). We evaluate models on the length and primitive splits of SCAN in this work.

Other synthetic datasets have been developed to evaluate aspects of compositional generalization beyond SCAN, including NACS (Bastings et al., 2018), CFQ (Keysers et al., 2019), and COGS (Kim and Linzen, 2020).

Another method for generating train and test splits is the template split (Finegan-Dollak et al., 2018). Unlike the aforementioned evaluations, template splits have been applied to non-synthetic datasets, primarily text-to-SQL datasets. In template splits, any parse template (defined as the target SQL query with entities anonymized) appearing in the training set cannot appear in the test set. While not a primary focus of our work, we analyze template splits and show that our TMCD splits can be significantly more challenging in Section 6.1.

In addition to introducing the CFQ dataset, Keysers et al. (2019) propose a method to formalize and generalize the procedures for generating compositional splits used by SCAN and template splits based on the notion of a compound distribution. Given this definition, they propose an algorithm for generating train and test splits that maximize the divergence of their respective compound distributions while bounding the divergence of their respective atom distributions. These splits are termed *Maximum Compound Divergence* (MCD) splits. We use the MCD split they generated for SCAN to evaluate models, and extend their methodology to create new TMCD splits for non-synthetic datasets in Section 3.

Finally, Herzig and Berant (2019) studies biases resulting from methods for efficiently collecting human-labeled data, providing further motivation for out-of-distribution evaluations.

Approaches Many specialized architectures have been developed to address the compositional generalization challenges of SCAN. Several of them have recently reached 100% accuracy across multiple SCAN challenges (Liu et al., 2020; Nye et al., 2020; Chen et al., 2020). Similarly to the NQG-T5 approach we propose in Section 4, all of these models incorporate discrete structure. However, unlike NQG-T5, they have only been evaluated on synthetic tasks.

Concurrently with this work, Herzig and Berant (2020) also begins to address our research question, proposing an approach that not only solves several SCAN challenges but also achieves strong performance on the standard and template splits of the non-synthetic dataset GeoQuery. However, their approach requires some manual task-specific engineering. We compare NQG-T5 with this approach and other SCAN-inspired architectures in Section 5.

The effect of large-scale pre-training on compositional generalization ability has also been studied. While giant pre-trained seq2seq models made significant improvements across many NLP tasks such as question answering (Raffel et al., 2019), Furrer et al. (2020) finds that they cannot solve several compositional generalization challenges.

While our work primarily focuses on modeling approaches, compositional data augmentation techniques have also been proposed (Jia and Liang, 2016; Andreas, 2019). NQG-T5 outperforms previously reported results for these methods, but we believe more work is needed to understand to what degree data augmentation can be a scalable and sufficient solution for improving compositional generalization.

Concurrently with this work, Oren et al. (2020) also explored compositional generalization on non-synthetic datasets, by focusing on the text-to-SQL datasets with template splits proposed by Finegan-Dollak et al. (2018). They propose several enhancements to seq2seq models that improve performance, but conclude that achieving strong performance on these settings remains an open challenge.

3 Target Maximum Compound Divergence (TMCD) Splits

We propose a new methodology for generating data splits that maximize compound divergence over non-synthetic datasets, which we call Target Maximum Compound Divergence (TMCD) splits. In standard MCD splits (Keysers et al., 2019), the notion of compounds requires that both source and targets are generated by a rule-based procedure, and therefore cannot be applied to existing non-synthetic datasets where natural language utterances are collected from humans.

For TMCD, we propose a new notion of compounds based only on the target representations. We leverage their known syntactic structure to define atoms and compounds. For instance, examples atoms in FunQL are `longest` and `river`, and an example compound is `longest(river)`. Detailed definitions of atoms and compounds for each dataset we study can be found in Appendix A.4.

Given this definition of compounds, our definition of compound divergence, $\mathcal{D}_C$, is the same as that of Keysers et al. (2019). Specifically,

$$\mathcal{D}_C = 1 \; - \; C_{0.1}(\mathcal{F}_{\text{TRAIN}} \,\|\, \mathcal{F}_{\text{TEST}}),$$

where $\mathcal{F}_{\text{TRAIN}}$ and $\mathcal{F}_{\text{TEST}}$ are the weighted frequency distributions of compounds in the training and test sets, respectively. The Chernoff coefficient $C_\alpha(P\|Q) = \sum_k p_k^\alpha \, q_k^{1-\alpha}$ (Chung et al., 1989) is used with $\alpha = 0.1$.

For TMCD, we constrain atom divergence by requiring that every atom appear at least once in the training set. A greedy algorithm similar to the one of Keysers et al. (2019) is used to generate splits that approximately maximize compound divergence. First, we randomly split the dataset. Then, we swap examples until the atom constraint is satisfied. Finally, we sequentially identify example pairs that can be swapped between the train and test set to increase compound divergence without violating the atom constraint, breaking when a swap can no longer be identified.

4 Proposed Approach: NQG-T5

Our proposed approach, NQG-T5, combines T5 (Raffel et al., 2019), a pre-trained seq2seq model, with a high-precision grammar-based component that we refer to as **NQG**, as it consists of a discriminative **N**eural parsing model and a **Q**uasi-synchronous **G**rammar induction algorithm (some of the induced grammar rules are illustrated in Table 1 and a detailed description of NQG is in Section 4.3).

Two common limitations of grammar-based approaches are the need for task-specific assumptions during grammar induction, and limited recall due to the rigidity of grammars. We address these limitations by using a general grammar induction algorithm that can operate over arbitrary pairs of strings, and leveraging the combination with T5 as a simple-yet-effective solution to limited recall.

4.1 Overview

We frame semantic parsing as translation between pairs of strings, where the source string $\mathbf{x}$ is a natural language utterance, and the target string $\mathbf{y}$ is a meaning representation or executable program (which we refer to as a target representation).

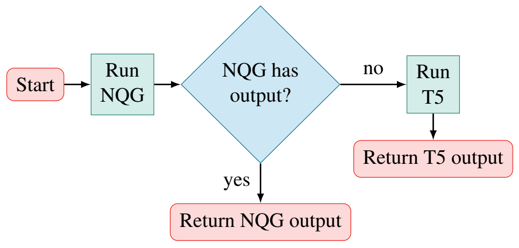

Figure 2: Overview of how predictions are generated by NQG-T5, a simple yet effective combination of T5 (Raffel et al., 2019) with a quasi-synchronous grammar induction algorithm and neural parser, NQG.

Let $\mathbf{x} = (x_1, \ldots, x_{|\mathbf{x}|})$ be a source string of $|\mathbf{x}|$ tokens and $\mathbf{y} = (y_1, \ldots, y_{|\mathbf{y}|})$ be a target string of $|\mathbf{y}|$ tokens. Our goal is to estimate $p(\mathbf{y} \mid \mathbf{x})$, the conditional probability of the target string $\mathbf{y}$ given the source string $\mathbf{x}$.

Inference For a given source string $\mathbf{x}$, a high-precision grammar-based approach such as NQG can fail to produce a derivation. Therefore, as demonstrated in Figure 2, we output the NQG output if one is generated, and output the T5 output otherwise. This simple combination can work well because NQG often has higher precision than T5 for cases where it produces an output, especially in out-of-distribution settings.

Training For a given training split, we train the T5 component and the NQG component separately, as described in the following sections.

4.2 T5 Component

T5 (Raffel et al., 2019) is a pre-trained sequence-to-sequence model based on the Transformer architecture (Vaswani et al., 2017). As our task formulation treats semantic parsing as a sequence-to-sequence task, it is straightforward to finetune a T5 model for our setting.

4.3 NQG Component

NQG combines a QCFG induction algorithm with a neural parsing model. Training is a two-stage process. First, we employ a compression-based grammar induction technique to construct our grammar $\mathcal{G}$ (Section 4.3.1). Second, based on the induced grammar, we build the NQG semantic parsing model via a discriminative latent variable model, using a powerful neural encoder to score grammar rule applications anchored in the source string $\mathbf{x}$ (Section 4.3.2).

SCAN
$NT \rightarrow \langle$turn right, I_TURN_RIGHT$\rangle$
$NT \rightarrow \langle NT_{[1]}$ after $NT_{[2]}, NT_{[2]}\ NT_{[1]}\rangle$
$NT \rightarrow \langle NT_{[1]}$ thrice, $NT_{[1]}\ NT_{[1]}\ NT_{[1]}\rangle$
GEOQUERY
$NT \rightarrow \langle$names of $NT_{[1]}, NT_{[1]}\rangle$
$NT \rightarrow \langle$towns, cities$\rangle$
$NT \rightarrow \langle NT_{[1]}$ have $NT_{[2]}$ running through them, intersection ($NT_{[1]}$, traverse_1 ($NT_{[2]}$))$\rangle$
SPIDER-SSP
$NT \rightarrow \langle$what is the id of the $NT_{[1]}$ named $NT_{[2]}$?, select rid from $NT_{[1]}$ where name = " $NT_{[2]}$ "$\rangle$

Table 1: Examples of QCFG rules generated by our induction algorithm. The subscript 1 in $NT_{[1]}$ indicates the correspondence between source and target non-terminals.

4.3.1 NQG Grammar Induction

We begin by introducing the grammar formalism we use. Compared to related work based on synchronous context free grammars (SCFG) for machine translation and semantic parsing, NQG uses a slightly more general grammar formalism that allows repetition of a non-terminal with the same index on the target side. We therefore adopt the terminology of QCFGs (Smith and Eisner, 2006), or Quasi-Synchronous Context-Free Grammars, to refer to our induced grammar $\mathcal{G}$.

Some examples of induced rules are shown in Table 1. Relevant background and notation for SCFGs and QCFGs and an example derivation (Figure 4) are provided in Appendix A.1.

Our grammar $\mathcal{G}$ contains a single non-terminal symbol, NT. We restrict source rules to ones containing at most 2 non-terminal symbols, and do not allow unary productions as source rules. This enables efficient parsing using an algorithm similar to CKY (Cocke, 1969; Kasami, 1966; Younger, 1967) that does not require binarization of the grammar.

To induce $\mathcal{G}$ from the training data, we propose a QCFG induction algorithm that does not rely on task-specific heuristics or separately derived word alignments. At a high-level, the grammar induction is guided by the principle of Occam's razor, which leads us to seek the smallest, simplest grammar that explains the data well.

We follow the Minimum Description Length (MDL) principle (Rissanen, 1978; Grunwald, 2004) as a way to formalize this intuition. Specifically, we use standard two-part codes to compute description length, where we are interested in an encoding of targets $\mathbf{y}$ given the inputs $\mathbf{x}$, across a dataset $\mathcal{D}$ consisting of these pairs. A two-part code encodes the model, and the targets encoded using the model; the two parts measure the simplicity of the model and the extent to which it can explain the data, respectively.

For grammar induction, our model is simply our grammar, $\mathcal{G}$. The codelength can therefore be expressed as $H(\mathcal{G}) - \sum_{\mathbf{x},\mathbf{y}\in\mathcal{D}} \log_2 P_{\mathcal{G}}(\mathbf{y}|\mathbf{x})$ where $H(\mathcal{G})$ corresponds to the codelength of some encoding of $\mathcal{G}$. We approximate $H(\mathcal{G})$ by counting terminal (C_T) and non-terminal (C_N) symbols in the grammar's rules, $\mathcal{R}$. For $P_{\mathcal{G}}$, we assume a uniform distribution over the set of targets that can be derived.[3] As the only mutable aspect of the grammar during induction is the set of rules $\mathcal{R}$, we abuse notation slightly and write our approximate codelength objective as a function of $\mathcal{R}$ only:

$$L(\mathcal{R}) = l_N C_N(\mathcal{R}) + l_T C_T(\mathcal{R}) + \sum_{\mathbf{x}\in\mathcal{D}} \log_2 C_\beta(\mathcal{R}, \mathbf{x}),$$

where $C_\beta(\mathcal{R}, \mathbf{x}) = |\{\beta | \langle NT, NT\rangle \overset{*}{\Rightarrow} \langle \mathbf{x}, \beta\rangle\}|$ is the number of unique targets that can be derived given each source, $\mathbf{x}$. l_N and l_T can be interpreted as the average bitlength for encoding non-terminal and terminal symbols, respectively. In practice, these can be treated as hyperparameters.

Having defined the codelength scoring function that we use to compare grammars, we describe our greedy search algorithm that finds a grammar that approximately minimizes this objective.

At initialization, we begin by creating a rule for every example in the dataset. The initial grammar fits the training data extremely well, but is also very large. We initialize $\mathcal{R}$ to be $\{NT \rightarrow \langle \mathbf{x}, \mathbf{y}\rangle \mid \mathbf{x}, \mathbf{y} \in \mathcal{D}\}$. We also add identity rules for substrings that exactly match between source and target examples, e.g. $NT \rightarrow \langle k, k\rangle$ where k is a substring of both $\mathbf{x}$ and $\mathbf{y}$ for some $\mathbf{x}, \mathbf{y} \in \mathcal{D}$. Our greedy algorithm iteratively identifies a rule to be added to $\mathcal{R}$ that maximizes $-\Delta L(\mathcal{R})$ by enabling ≥ 1 rules in $\mathcal{R}$ to be removed while maintaining the invariant that $\mathcal{G}$ allows for deriving all of the training examples, i.e. $\langle NT, NT\rangle \overset{*}{\Rightarrow} \langle \mathbf{x}, \mathbf{y}\rangle$ for

[3]This can be viewed as a conservative choice, as in practice our neural parser would learn a better model for $P(\mathbf{y}|\mathbf{x})$ than the naive uniform encoding over possible targets.

every $\mathbf{x}, \mathbf{y} \in \mathcal{D}$. The search completes when no rule that decreases $L(\mathcal{R})$ can be identified.

In practice, we use several approximations to efficiently select a rule at each iteration. Additional details are described in Appendix A.2.

4.3.2 NQG Semantic Parsing Model

Based on the inducted grammar $\mathcal{G}$, we describe our neural semantic parsing model as follows. Let $Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}$ be the set of all derivations in $\mathcal{G}$ that yield the pair of strings $\mathbf{x}$ and $\mathbf{y}$. And let $Z^{\mathcal{G}}_{\mathbf{x},*} \supset Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}$ be the set of derivations for $\mathbf{x}$ and any target string.

We define $p(\mathbf{y} \mid \mathbf{x})$ as the marginalization over all derivations that produce $\mathbf{y}$ given $\mathbf{x}$:

$$p(\mathbf{y} \mid \mathbf{x}) = \sum_{z \in Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}} p(\mathbf{z} \mid \mathbf{x}),$$

where $\mathbf{z}$ is a QCFG derivation. We model $p(\mathbf{z} \mid \mathbf{x})$ using the same formulation as the Neural CRF model of Durrett and Klein (2015):

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{\exp(s(\mathbf{z}, \mathbf{x}))}{\sum\limits_{z' \in Z^{\mathcal{G}}_{\mathbf{x},*}} \exp(s(\mathbf{z}', \mathbf{x}))},$$

where $s(\mathbf{z}, \mathbf{x})$ is a derivation score and the denominator is a global partition function.

The scores decompose over anchored rules from our grammar:

$$s(\mathbf{z}, \mathbf{x}) = \sum_{(r,i,j) \in \mathbf{z}} \phi(r, i, j, \mathbf{x}),$$

where r is an index for a rule in $\mathcal{G}$ and i and j are indices defining the anchoring in $\mathbf{x}$. The anchored rule scores, $\phi(r, i, j, \mathbf{x})$, are based on contextualized representations from a BERT (Devlin et al., 2018) encoder:

$$\phi(r, i, j, \mathbf{x}) = f_s([w_i, w_j]) + e_r^{\intercal} f_r([w_i, w_j]),$$

where $[w_i, w_j]$ is the concatenation of the BERT representations for the first and last wordpiece in the anchored span, f_r is a feed-forward network with hidden size d that outputs a vector $\in \mathbb{R}^d$, f_s is a feed-forward network with hidden size d that outputs a scalar, and e_r is an embedding $\in \mathbb{R}^d$ for the rule index r. This formulation for encoding spans is similar to that used in other neural span-factored models (Stern et al., 2017).

At training time, we use a Maximum Marginal Likelihood (MML) objective, $-\log(P(\mathbf{y}|\mathbf{x}))$. We pre-process each example to produce a parse forest representation for both $Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}$ and $Z^{\mathcal{G}}_{\mathbf{x},*}$, which correspond to the numerator and denominator of our MML objective, respectively. This enables us to efficiently sum derivation scores over these static parse forest structures inside the training loop.

At inference time, we select the highest scoring derivation using an algorithm similar to CKY that considers anchored rule scores generated by the neural parsing model. We output the corresponding target if it can be derived by a CFG defining valid target constructions for the given task.

4.3.3 NQG Discussion

NQG is inspired by more traditional approaches to semantic parsing based on grammar formalisms such as CCG (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010, 2013) and SCFG (Wong and Mooney, 2006, 2007; Andreas et al., 2013; Li et al., 2015).

These formalisms directly embody the formal notion of the principle of compositionality as a homomorphism between the parts of the source and target structures, and operations over these parts (Montague, 1970; Janssen and Partee, 1997).

NQG is also closely related to work that uses synchronous grammars for hierarchical statistical machine translation, such as Hiero (Chiang, 2007). Unlike Hiero, NQG does not rely on an additional word alignment component. Moreover, Hiero simply uses relative frequency to learn rule weights. Our parsing model uses a discriminative latent variable training method similar to that of Blunsom et al. (2008), with the difference that we employ a neural instead of a log-linear derivation scoring model.

Unlike traditional SCFG models for machine translation applied to semantic parsing (Wong and Mooney, 2006; Andreas et al., 2013), our neural model conditions on global context from the source $\mathbf{x}$ via contextual word embeddings, and our grammar's rules do not need to carry source context to aid disambiguation.

Our grammar induction algorithm is similar to the inversion transduction grammar induction method for machine translation by Saers et al. (2013). More broadly, compression-based criteria have been successfully used by a variety of models for language (Grünwald, 1995; Ravi and Knight, 2009; Poon et al., 2009).

5 Experiments

We evaluate NQG-T5 and several strong baselines across a diverse set of evaluations to assess both compositional generalization and handling of natural language variation.

First, we compare the approaches on various challenging splits of two datasets with compositional queries: SCAN (Lake and Baroni, 2018) and GEOQUERY (Zelle, 1995). Both datasets are closed-domain and have outputs with straightforward syntax, enabling us to make clear comparisons between synthetic vs. non-synthetic setups.

Second, we evaluate the approaches on SPIDER (Yu et al., 2018), a non-synthetic text-to-SQL dataset, which includes compounding challenges such as schema linking and handling of complex SQL syntax.

5.1 Experiments on SCAN and GEOQUERY

Approaches to compare We primarily evaluate the NQG-T5 model proposed in Section 4. To assess the effect of model size, we compare two different sizes of the underlying T5 model: Base (220 million parameters) and 3B (3 billion parameters). We also evaluate the NQG component individually, treating any example where no output is provided as incorrect when computing accuracy.

We compare NQG-T5 to two families of baseline approaches described in Figure 1. First, for general-purpose models, we consider T5 (Raffel et al., 2019) in both Base and 3B sizes. T5 is a large pre-trained seq2seq model which achieves state-of-the-art results across many NLP tasks.

Second, for specialized methods with strong compositional bias, we consider approaches that have been developed for SCAN. Some previous works on SCAN require task-specific information such as the mapping of atoms (Lake, 2019; Gordon et al., 2019) or a grammar mimicking the training data (Nye et al., 2020), and as such are difficult to adapt to non-synthetic datasets. Among the works that do not need task-specific resources, we evaluate the two models with publicly available code: Syntactic Attention (Russin et al., 2019) and CGPS (Li et al., 2019). We report results on SCAN from the original papers as well as new results on our proposed data splits.

Datasets We compare the approaches on the SCAN and GEOQUERY datasets. For SCAN, we evaluate using the length split and two primitive splits, *jump* and *turn left*, included in the original dataset (Lake and Baroni, 2018). We also evaluate using the SCAN MCD splits from Keysers et al. (2019).

The GEOQUERY dataset (Zelle, 1995) consists of natural language questions about US geography along with corresponding target representations. Similarly to prior work (Dong and Lapata, 2016, 2018), we replace entity mentions with placeholders (e.g., "`m0`", "`m1`") in both the source and target. We use a variant of Functional Query Language (FunQL) as the target representation (Kate et al., 2005). Besides the standard split of Zettlemoyer and Collins (2005), we also generate a new split based on query length, and a TMCD split, each consisting of 440 train and 440 test examples. We report exact-match accuracy for both datasets.

Hyperparameters and detailed pre-processing for all experiments can be found in Appendix A.3.

Results The results are presented in Table 2. We report metrics from corresponding papers for the baselines (gray cells). The results for T5 on SCAN are from Furrer et al. (2020). Additionally, we include the metrics from GECA[4] (Andreas, 2019), a data augmentation method, as well as LANE (Liu et al., 2020) and NSSM (Chen et al., 2020), two approaches on SCAN whose code is yet to be released. We also compare with the concurrently developed SpanBasedSP model[5] (Herzig and Berant, 2020).

From the results, we observe that the gains on compositional splits of SCAN do not necessarily correlate with the results on GEOQUERY. Figure 3 plots the accuracy of different approaches on the (T)MCD splits of the two datasets. T5 struggles on the SCAN dataset, while the two SCAN baselines have low accuracy on both of these challenging splits. In contrast, the proposed NQG-T5 approach combines the strengths of T5 and NQG to achieve good results on both splits.

Finally, to the best of our knowledge, both T5 and NQG-T5 achieve new state-of-the-art accuracy on the standard split of GEOQUERY.

[4]GECA reports GEOQUERY results on a setting with Prolog logical forms and without anonymization of entities. Note that the performance of GECA depends on both the quality of the generated data and the underlying parser (Jia and Liang, 2016), which can complicate the analysis.

[5]SpanBasedSP also reports GEOQUERY results using FunQL, but uses different data preprocessing, only 540 examples for training, and reports denotation accuracy. For SCAN, they pre-process the dataset to produce program-level supervision.

System	SCAN Jump	SCAN Turn Left	SCAN Len	SCAN MCD	GeoQuery Standard	GeoQuery Len	GeoQuery TMCD
LANE (Liu et al., 2020)	**100**	—	**100**	**100**	—	—	—
NSSM (Chen et al., 2020)	**100**	—	**100**	—	—	—	—
Syntactic Attention (Russin et al., 2019)	91.0	**99.9**	15.2	2.9	77.5	23.6	0.0
CGPS (Li et al., 2019)	98.8	**99.7**	20.3	2.0	62.1	9.3	32.3
GECA (Andreas, 2019)	87.0	—	—	—	78.0[†]	—	—
SBSP (Herzig and Berant, 2020)	**100**	**100**	**100**	**100**	86.1[†]	—	—
SBSP −*lexicon*	**100**	**100**	**100**	**100**	78.9[†]	—	—
T5-Base (Raffel et al., 2019)	**99.5**	62.0	14.4	15.4	**92.9**	39.1	54.3
T5-3B (Raffel et al., 2019)	99.0	65.1	3.3	11.6	**93.2**	36.8	51.6
NQG-T5-Base	**100**	**100**	**100**	**100**	92.5	**50.9**	**56.6**
NQG-T5-3B	**100**	**100**	**100**	**100**	**93.2**	50.2	54.1
NQG	**100**	**100**	**100**	**100**	73.9	35.0	40.7

Table 2: **Main Results.** Existing approaches have yet to excel on a diverse set of evaluations across synthetic and non-synthetic tasks, while NQG-T5 obtains significant improvement. Gray cells are previously reported results. [†] indicates differences in GeoQuery settings (see discussion in Section 5.1). Boldfaced results are within 0.5 points of the best result.

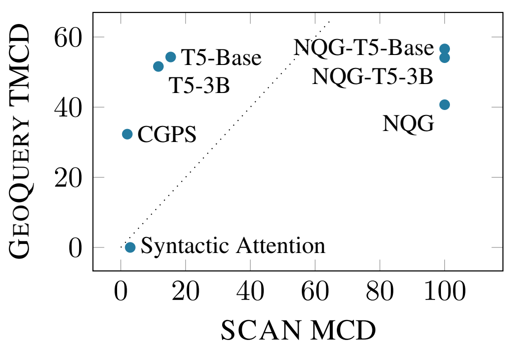

Figure 3: We plot performance on GeoQuery TMCD compared to SCAN MCD, which show relatively low correlation. Dotted line is equal performance. Improvements on SCAN do not necessarily translate to improvements on non-synthetic data.

5.2 Experiments on Spider

Spider is a text-to-SQL dataset that consists of 10,181 questions and 5,693 unique complex SQL queries across 138 domains (Yu et al., 2018). The primary evaluation for Spider is in the cross-database setting, where models are evaluated on examples for databases not seen during training. One of the primary challenges in this setting is generalization to new database schemas, while in this work we aim to focus more on out-of-distribution compositional generalization.

Therefore, we use a setting similar to an alternative setting called the example split in the original dataset (Yu et al., 2018) where the databases are shared between train and test examples. We identify examples in the original train set for databases with more than 50 examples to ensure sufficient coverage over table and column names in the training data. We then generate 3 new train and test splits consisting of 3,282 train and 1,094 test examples across 51 databases: a random split, a split based on source length, and a TMCD split. We adopt the terminology of Suhr et al. (2020) and use Spider-SSP to refer to these same-database splits, and use Spider-XSP to refer to the standard cross-database setting.

We prepend the name of the target database to the source sequence. For T5, we also serialize the database schema as a string and append it to the source sequence similar to Suhr et al. (2020). We report exact set match without values, the standard Spider evaluation metric (Yu et al., 2018).

Results Table 3 shows the results of T5 and NQG-T5 on different splits of Spider-SSP. We also show T5-Base performance without the schema string appended. The text-to-SQL mapping is not well modeled by NQG: SQL has complicated syntax and often requires complex coordination across discontinuous clauses. Intermediate representations for SQL such as SemQL (Guo et al., 2019) may help increase the correspondence between source and target syntax. Nevertheless, the performance of NQG-T5 is competitive with T5, indicating a strength of the hybrid approach.

Table 4 shows the results on Spider-XSP, which focuses primarily on handling unseen

System	Spider-SSP Random	Len	TMCD
T5-Base −*schema*	76.5	42.5	42.3
T5-Base	82.0	49.0	60.9
T5-3B	85.6	56.7	69.6
NQG-T5-Base	81.8	49.0	60.8
NQG-T5-3B	85.4	56.7	69.5
NQG	1.3	0.0	0.5

Table 3: Results on Spider-SSP. Although the text-to-SQL task is not modeled well by the NQG grammar due to SQL's complex syntax, NQG-T5 is still able to perform well by relying on T5. Note that the length and TMCD splits are significantly more challenging than random splits.

System	Spider-XSP Dev
RYANSQL v2 (Choi et al., 2020)	70.6
RATSQL v3 (Wang et al., 2020)	69.7
T5-Base	57.1
T5-3B	70.0
NQG-T5-Base	57.1
NQG-T5-3B	70.0
NQG	0.0

Table 4: Although Spider-XSP is not our focus, T5-3B and NQG-T5-3B are competitive with the state-of-the-art, without any model modifications for the cross-database setting.

schema rather than compositional generalization. To our surprise, the T5-3B model proves to be competitive with the state-of-the-art (Choi et al., 2020; Wang et al., 2020) without model modifications or access to database contents beyond the table and column names. As NQG-T5 simply uses T5's output when the induced grammar lacks recall, it too is competitive.

6 Analysis

6.1 Comparison of Data Splits

Table 6 compares the compound divergence, the number of test examples with unseen atoms, and the accuracy of T5-Base across various splits. In addition to the splits from our experiments, we also include an even random split of GeoQuery (440 train and 440 test examples) and a template split (Finegan-Dollak et al., 2018) for GeoQuery based on FunQL logical forms (441 training and 439 test examples). While template splits can still be useful evaluations, TMCD splits appear to be more challenging, despite having stronger constraints to avoid unseen atoms at test time.

Length splits are also very challenging, but they primarily target a more specific aspect of compositional generalization, leading to more predictable error patterns for seq2seq models.

6.2 T5 Analysis

We now analyze each component of NQG-T5, starting with T5.

On length splits, there is a consistent pattern to the errors. T5's generated target sequences on the test set are generally not significantly longer than the maximum length observed during training, leading to poor performance when gold sequences are significantly longer. This phenomena was explored in detail by Newman et al. (2020) concurrently with this work.

Diagnosing the large generalization gap on the (T)MCD splits is more challenging, but we noticed several error patterns. For T5-Base on the GeoQuery TMCD split, in 52 of the 201 incorrect predictions (26%), the first incorrectly predicted symbol occurs when the gold symbol has 0 probability under a trigram language model fit to the training data. This suggests that the decoder's implicit target language model might have over-fitted to the distribution of target sequences in the training data, hampering its ability to generate novel compositions. In other cases, the errors appear to reflect over-fitting to spurious correlations between source and target strings, rather than learning the correct underlying compositional structure of the mapping.

6.3 NQG Analysis

To analyze NQG, we compute its coverage (fraction of examples where NQG produces an output) and precision (fraction of examples with a correct output among ones where an output is produced) on different data splits. The results in Table 5 show that NQG has high precision but struggles at recall on some data splits.

There is a significant difference between the effectiveness of the grammar induction procedure among the three datasets, as shown in Table 7. Unsurprisingly, grammar induction is most effective for SCAN, where the data generating process has similar expressive power to a QCFG. We hypothesize that, generally, synthetic datasets may be more amenable to compression via symbolic methods than non-synthetic datasets, as the data

NQG	SCAN				GeoQuery			Spider-SSP			Spider-XSP
Metric	Jump	Turn Left	Len	MCD	Standard	Len	TMCD	Random	Len	TMCD	Dev
Coverage	100	100	100	100	77.1	40.0	43.0	1.5	0.0	0.6	0.0
Precision	100	100	100	100	95.8	87.5	94.7	87.5	—	85.7	—

Table 5: NQG Coverage (% of examples where an output is produced) and precision (% of cases where the output is correct when one is generated). NQG-T5 outperforms T5 when NQG has higher precision than T5 over the subset of examples it covers.

Dataset	Split	Missing $\%_A$	$\mathcal{D}_C$	T5-Base
GeoQuery	Standard	0.3	0.03	92.9
GeoQuery	Random	1.4	0.03	91.1
GeoQuery	Template	0.9	0.07	87.0
GeoQuery	Length	4.3	0.17	39.1
GeoQuery	TMCD	0	0.19	54.3
Spider-SSP	Random	6.2	0.03	82.0
Spider-SSP	Length	27.4	0.08	49.0
Spider-SSP	TMCD	0	0.18	60.9

Table 6: Percentage of test examples with atoms not included in the training set (Missing $\%_A$), compound divergence ($\mathcal{D}_C$), and T5-Base accuracy for various dataset splits. Note that Length Splits could have high percentage of missing atoms in the training set.

Dataset	Examples	Induced Rules	Ratio
SCAN	16727	20	836.4
GeoQuery	600	234	2.6
Spider-SSP	3282	4155	0.79

Table 7: Size of the original dataset and the number of rules in the induced QCFG grammar for a selected split from each task. Synthetically-generated datasets may be more amenable to compression via symbolic methods, as they are often generated by relatively short symbolic programs.

generating procedure is typically a relatively short program. QCFG induction is particularly unsuccessful for Spider, where the resulting number of rules is larger than the original dataset due to the identity rules added during initialization. The complex syntax of SQL is not well modeled by QCFGs, and induction fails to induce rules that enable significant generalization. Most of the induced rules are limited to simply replacing table and column names or value literals with non-terminals, such as the rule shown in Table 1, rather than inducing rules representing nested substructures.

For both GeoQuery and Spider, NQG is limited by the expressiveness of QCFGs and the simple greedy search procedure used for grammar induction, which can lead to sub-optimal approximations of the induction objective. Notably, QCFGs cannot represent relations between source strings, such as semantic similarity, or relations between target strings, such as logical equivalence (e.g. `intersect(a,b)` $\Leftrightarrow$ `intersect(b,a)`), that could enable greater generalization. However, such extensions pose additional scalability challenges, requiring new research in more flexible approaches for both learning and inference.

7 Conclusions

Our proposed approach, NQG-T5, improves performance across several evaluations of out-of-distribution compositional generalization while also being competitive with the state-of-the-art on standard evaluations. While we found that the grammar-based component of NQG-T5 is capable of high precision even in out-of-distribution settings, it also has low coverage for some tasks, especially those with complex syntax such as Spider. Expanding the coverage of such an approach while maintaining high precision is an important future direction. More broadly, our work highlights that evaluating on a diverse set of benchmarks is important and that handling both out-of-distribution compositional generalization and natural language variation remains an open challenge for semantic parsing.

8 Acknowledgements

We thank Kenton Lee, William Cohen, Jeremy Cole, and Luheng He for helpful discussions. Thanks also to Emily Pitler and Jonathan Herzig for their comments.

References

Alfred V Aho and Jeffrey D Ullman. 1972. *The theory of parsing, translation, and compiling*, volume 1.

Prentice-Hall Englewood Cliffs, NJ.

Jacob Andreas. 2019. Good-enough compositional data augmentation. *arXiv preprint arXiv:1904.09545*.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria. Association for Computational Linguistics.

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: Scan both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio. Association for Computational Linguistics.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. *arXiv preprint arXiv:2008.06662*.

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

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases.

JK Chung, PL Kannappan, CT Ng, and PK Sahoo. 1989. Measures of distance between probability distributions. *Journal of mathematical analysis and applications*, 138(1):280–292.

John Cocke. 1969. *Programming languages and their compilers: Preliminary notes*. New York University.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029*.

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*.

Peter Grünwald. 1995. A minimum description length approach to grammar inference. In *International Joint Conference on Artificial Intelligence*, pages 203–216. Springer.

Peter Grunwald. 2004. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535.

Jonathan Herzig and Jonathan Berant. 2019. Don't paraphrase, detect! rapid and effective data collection for semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3801–3811.

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

Theo MV Janssen and Barbara H Partee. 1997. Compositionality. In *Handbook of logic and language*, pages 417–473. Elsevier.

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.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Rohit J Kate, Yuk Wah Wong, and Raymond J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1062. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1545–1556.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics.

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.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, pages 9788–9798.

Brenden M Lake, Tal Linzen, and Marco Baroni. 2019. Human few-shot learning of compositional instructions. *arXiv preprint arXiv:1901.04587*.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Junhui Li, Muhua Zhu, Wei Lu, and Guodong Zhou. 2015. Improving semantic parsing with enriched synchronous context-free grammar. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1465, Lisbon, Portugal. Association for Computational Linguistics.

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

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020. Compositional generalization by learning analytical expressions. *arXiv preprint arXiv:2006.10627*.

Richard Montague. 1970. Universal grammar. *Theoria*, 36(3):373–398.

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D Manning. 2020. The eos decision and length extrapolation. *arXiv preprint arXiv:2010.07174*.

Maxwell I Nye, Armando Solar-Lezama, Joshua B Tenenbaum, and Brenden M Lake. 2020. Learning compositional rules via neural program synthesis. *arXiv preprint arXiv:2003.05562*.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. *arXiv preprint arXiv:2010.05647*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado. Association for Computational Linguistics.

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*.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512, Suntec, Singapore. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

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*.

Markus Saers, Karteek Addanki, and Dekai Wu. 2013. Unsupervised transduction grammar induction via minimum description length. In *Proceedings of the Second Workshop on Hybrid Approaches to Translation*, pages 67–73, Sofia, Bulgaria. Association for Computational Linguistics.

David A 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.

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

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8372–8388.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446. Association for Computational Linguistics.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

John Marvin Zelle. 1995. *Using inductive logic programming to automate the construction of natural language parsers*. Ph.D. thesis, Citeseer.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666. AUAI Press.

A Supplemental Material

A.1 Background: QCFGs and SCFGs

Synchronous Context-Free Grammars (SCFGs) have been used to model the hierarchical mapping between pairs of strings in areas such as compiler theory (Aho and Ullman, 1972) and natural language processing.

Informally, SCFGs can be viewed as an extension of Context-Free Grammars (CFGs) that *synchronously* generate strings in both a source and target language. We write SCFG rules as:

$$S \rightarrow \langle \alpha, \beta \rangle$$

Where S is a non-terminal symbol, and α and β are strings of non-terminal and terminal symbols. An SCFG rule can be viewed as two CFG rules, $S \rightarrow \alpha$ and $S \rightarrow \beta$ with a pairing between the occurrences of non-terminal symbols in α and β. This pairing is indicated by assigning each non-terminal in α and β an index $\in \mathbb{N}$. Non-terminals sharing the same index are called *linked*. Following convention, we denote the index for a non-terminal using a boxed subscript, e.g. $NT_{\boxed{1}}$.

A complete SCFG derivation is a pair of parse trees, one for the source language and one for the target language. For our purposes, we are primarily interested in translation with a SCFG, where given a string in the source language, $\mathbf{x}$, and an SCFG, G, we are interested in generating derivations that produce a pair of strings $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{y}$ is some string from the target language.

An example derivation is shown in figure 4.

Quasi-Synchronous Context-Free Grammars (QCFGs) QCFGs generalize SCFGs in various ways, notably relaxing the restriction on a strict one-to-one alignment between source and target non-terminals (Smith and Eisner, 2006). This can allow for modeling phenomena such as a one-to-many mapping between source and target constituents.

Rule Application in QCFGs The $\Rightarrow^r$ operator refers to a *derives* relation, such that $\langle \alpha^1, \beta^1 \rangle \Rightarrow^r \langle \alpha^2, \beta^2 \rangle$ states that the string pair $\langle \alpha^2, \beta^2 \rangle$ can be generated from $\langle \alpha^1, \beta^1 \rangle$ by applying the rule r. We write $\Rightarrow$ to leave the rule unspecified, assuming the set of possible rules is clear from context. We write $\Rightarrow\Rightarrow$ to indicate a chain of 2 rule applications, omitting the intermediate string pair. Finally, we write $\overset{*}{\Rightarrow}$ to denote the reflexive transitive closure of $\Rightarrow$.

A.2 QCFG Induction Details

In this section we provide more details on the induction algorithm introduced in section 4.3.1. Our induction algorithm was designed with simplicity in mind, and therefore uses a simple greedy search process that could likely be significantly improved upon by future work. We sketch the algorithm below, noting where approximations are used in practice.

First, let us define several operations over rules and sets of rules. We define the set of rules that can be derived from a given set of rules, $\mathcal{R}$:

$$d(\mathcal{R}) = \{NT \rightarrow \langle \alpha, \beta \rangle | \langle NT, NT \rangle \overset{*}{\Rightarrow} \langle \alpha, \beta \rangle\}$$

We define an operation SPLIT that *splits* a rule into 2 rules:

$$\begin{gathered} \text{SPLIT}(NT \rightarrow \langle \alpha, \beta \rangle) = \{g, h | \\ \langle NT, NT \rangle \Rightarrow^g \Rightarrow^h \langle \alpha, \beta \rangle \vee \\ \langle NT, NT \rangle \Rightarrow^h \Rightarrow^g \langle \alpha, \beta \rangle\} \end{gathered}$$

Where $g, h \in \mathcal{R}$ is a pair of new rules that would maintain the invariant that $\langle NT, NT \rangle \overset{*}{\Rightarrow} \langle \mathbf{x}, \mathbf{y} \rangle$ for every $\mathbf{x}, \mathbf{y} \in \mathcal{D}$, even if the provided rule is eliminated.

SPLIT can be implemented by considering pairs of sub-strings of α and β to replace with a new indexed non-terminal symbol. For example, the rule $NT \rightarrow$ $\langle$largest state, largest (state)$\rangle$ can be split into the rules $NT \rightarrow \langle$largest $NT_{\boxed{1}}$, largest ($NT_{\boxed{1}}$)$\rangle$ and $NT \rightarrow \langle$state, state$\rangle$. This step can require re-indexing of non-terminals when the rule being split contains non-terminals.

We optionally allow SPLIT to introduce repeated target non-terminals when the target string has repeated substrings. Otherwise, we do not allow SPLIT to replace a repeated substring with a non-terminal, as this can lead to an ambiguous choice. We enable this option for SCAN and SPIDER but not for GEOQUERY, as FunQL does not require such repetitions.

While we do not fully leverage the known syntactic structure of target strings to restrict which substrings can be considered by SPLIT, we do require that any explicit bracketing given by parentheses is respected, i.e. that parentheses are balanced for all target strings.

During our greedy search, we only split rules when one of the two resulting rules can already be derived given $\mathcal{R}$. Therefore, we define a function

$NT \rightarrow \langle$how many $NT_{[1]}$ pass through $NT_{[2]}$, answer (count (intersection ($NT_{[1]}$, loc_1 ($NT_{[2]}$)))) $\rangle$
$NT \rightarrow \langle$rivers, river$\rangle$
$NT \rightarrow \langle$the largest $NT_{[1]}$, largest ($NT_{[1]}$)$\rangle$
$NT \rightarrow \langle$state, state$\rangle$

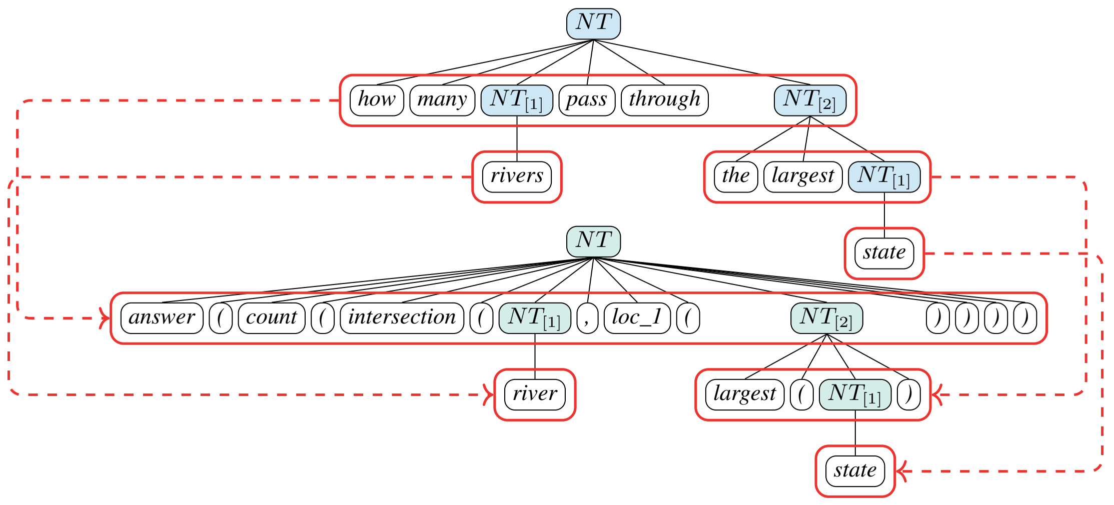

Figure 4: An example QCFG derivation. Each non-terminal node in the source derivation (blue) corresponds to a non-terminal in the target derivation (green). The QCFG rules used in the derivation are shown above.

NEW that returns a set of candidate rules to consider:

$$\text{NEW}(\mathcal{R}) = \\ \{g | g, h \in \text{SPLIT}(f) \wedge f \in \mathcal{R} \wedge h \in d(\mathcal{R})\}$$

Similarly, we can compute the set of rules that are redundant and can be eliminated by introducing one these candidate rules, f, as these rules can now be derived given the addition of f:

$$\text{ELIM}(\mathcal{R}, f) = \\ \{h | f, g \in \text{SPLIT}(h) \wedge g \in d(\mathcal{R}) \wedge h \in \mathcal{R}\}$$

We can then define the codelength reduction of adding a particular rule, $\Delta L(\mathcal{R}, f) = L(\mathcal{R}) - L(\mathcal{R}')$ where $\mathcal{R}' = (\mathcal{R} \cup f) \setminus \text{ELIM}(\mathcal{R}, f)$.

The codelength is defined in section 4.3.1. For all experiments, we use $l_N = 1$ and $l_T = 8$. The last term of the codelength is related to the increase in the count of spurious derivations due to introducing f, as described in 4.3.1. Rather than computing this exactly, we estimate this quantity by sampling up to k examples from $\mathcal{D}$ that contain all of the sub-strings of source terminal symbols in f such that f could be used in a derivation, and computing the increase in spurious derivations over this sample only. We then estimate the total increase by scaling according to the total number of examples that met the matching criteria. We sample $k = 10$ examples for all experiments.

Finally, we can select the rule with the largest ΔL:

$$\text{MAX}_{\Delta L}(\mathcal{R}) = \operatorname*{argmax}_{f \in \text{NEW}(\mathcal{R})} \Delta L(\mathcal{R}, f)$$

Conceptually, after initializing with rules based on the training data and adding identity rules, as discussed in section 4.3.1, the algorithm then proceeds as:

```
while $|\text{NEW}(\mathcal{R})| > 0$ do
    $r \leftarrow \text{MAX}_{\Delta L}(\mathcal{R})$
    if $\Delta L < 0$ then
        break
    end if
    $\mathcal{R} \leftarrow (\mathcal{R} \cup r) \setminus \text{ELIM}(\mathcal{R}, r)$
end while
```

For efficiency, we select the shortest N examples from the training dataset, and only consider these during the induction procedure. Avoiding longer examples is helpful as the number of candidates returned by SPLIT is polynomial with respect to source and target length. Once induction has completed, we then determine which of the longer examples cannot be derived based on the set of induced rules, and add rules for these examples. We use $N = 500$ for SCAN and $N = 1000$ for SPIDER. As the GEOQUERY training set contains < 500 unique examples, we use the entire training set.

Our algorithm maintains a significant amount of state between iterations to cache computations that are not affected by particular rule changes, based on overlap in terminal symbols.

We developed the algorithm and selected some hyperparameters by assessing the size of the induced grammars over the training sets of SCAN and GEOQUERY.

A.3 Experimental Configuration

A.3.1 Model Hyperparameters

We selected reasonable hyperparameter values and performed some minimal hyperparameter tuning for T5 and NQG based on random splits of the training sets for GEOQUERY and SPIDER. We used the same hyperparameters for all splits of each dataset without additional tuning.

For T5, we selected a batch size of 128 and then selected a learning rate of $1e^{-4}$ from $[1e^{-3}, 1e^{-4}, 1e^{-5}]$, which we used for all experiments. We fine-tune for $3,000$ steps for GEOQUERY and $10,000$ for SPIDER.

For the NQG neural model, for all experiments, we use the tiny pre-trained BERT model (Turc et al., 2019), with $d = 256$ dimensions for computing anchored rule scores. We fine-tune for 256 steps and use a learning rate of $1e^{-4}$.

A.3.2 Dataset Preprocessing

SCAN SCAN did not require any dataset-specific preprocessing.

GEOQUERY We use the version of the dataset with variable-free FunQL logical forms (Kate et al., 2005), and expand certain functions based on their logical definitions, such that `state(next_to_1(state(all)))` becomes the more conventional `intersection(state, next_to_1(state))`. We replace entity mentions with placeholders (e.g. "`m0`", "`m1`") in both the source and target.

SPIDER We prepend the name of the target database to the source sequence. For T5, we also serialize the database schema as a string and append it to the source sequence similarly to Suhr et al. (2020). This schema string contains the names of all tables in the database, and the names of the columns for each table. As we use a maximum source sequence length of 512 for T5, this leads to some schema strings being truncated (affecting about 5% of training examples).

A.4 Atom and Compound Definitions

GEOQUERY FunQL (Kate et al., 2005) targets consist of nested function trees. The tree structure is given by explicit bracketing. We define atoms as individual FunQL symbols, and compounds as combinations between parent and child symbols in the FunQL tree. Example atoms are `longest`, `river`, and `exclude` and example compounds are `longest(river)` and `exclude(longest(_), _)`.

SPIDER For SQL, we tokenize the SQL string and define atoms as individual tokens. To define compounds, we parse the SQL string using an unambiguous CFG, and define compounds from the resulting parse tree. We define compounds over both first and second order edges in the resulting parse tree.