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 approaches do not 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 on non-synthetic data, 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.

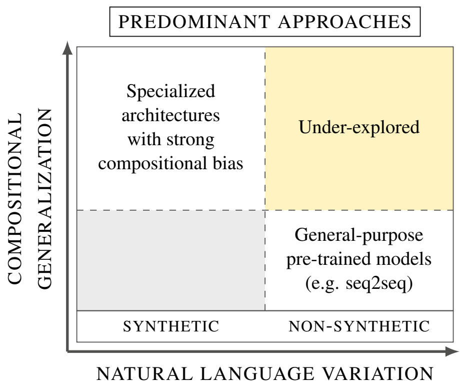
PREDOMINANT APPROACHES
Specialized architectures with strong compositional bias
Under-explored
General-purpose pre-trained models (e.g. seq2seq)
SYNTHETIC
NON-SYNTHETIC
COMPOSITIONAL GENERALIZATION
NATURAL LANGUAGE VARIATION

Figure 1: We study whether a semantic parsing approach can handle both out-of-distribution compositional generalization and natural language variation. Existing approaches are commonly evaluated across only one dimension.

1 Introduction

Sequence-to-sequence (seq2seq) models 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 often 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 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 approaches have only been evaluated on synthetic datasets. While synthetic datasets enable precise, interpretable evaluation of specific phenomena, they are 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?*

[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.

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. For example, large pre-trained seq2seq models that perform well on in-distribution evaluations do not address most of the 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). Therefore, we are interested in approaches that do well not only on controlled synthetic challenges of compositionality or in-distribution natural utterances, but across all of the diverse set of evaluations shown in Figure 2.

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. In addition, strong existing approaches do not perform well across all evaluations in Figure 2. We also propose new Target Maximum Compound Divergence (TMCD) train and test splits, extending the methodology of Keysers et al. (2020) to create challenging evaluations of compositional generalization for non-synthetic datasets. We show that TMCD splits complement existing evaluations by focusing on different aspects of the problem.

Second, on the modeling front, we propose NQG, a simple and general grammar-based approach that solves SCAN and also scales to natural utterances, obtaining high precision for non-synthetic data. In addition, we introduce and evaluate NQG-T5, a hybrid model that combines NQG with T5 (Raffel et al., 2020), leading to improvements across several compositional generalization evaluations while also being competitive on the standard splits of GeoQuery (Zelle and Mooney, 1996) and Spider (Yu et al., 2018). Our results indicate that NQG-T5 is a strong baseline for our challenge of developing approaches that perform well across a diverse set of evaluations focusing on either natural language variation, compositional generalization, or both. Comparing five approaches across eight evaluations on SCAN and GeoQuery, its average rank is 1, with the rank of the best previous approach (T5) being 2.9; performance is also competitive across several evaluations on Spider.

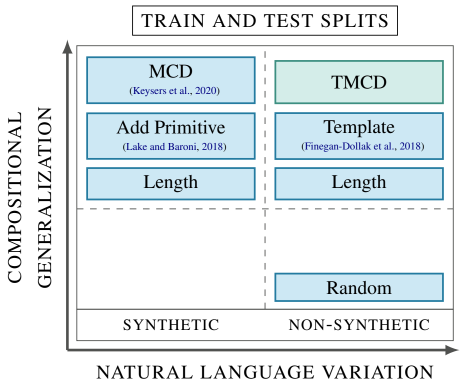

Figure 2: We evaluate semantic parsing approaches across a diverse set of evaluations focused on natural language variation, compositional generalization, or both. We add TMCD splits to complement existing evaluations. Ordering within each cell is arbitrary.

While still far from affirmatively answering our research 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.[3]

2 Background and Related Work

In this section, we survey recent work related to 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 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., "jump twice") mapping to action sequences (e.g., "`I_JUMP I_JUMP`"). One split for SCAN is the length split, where examples are separated by length such that the test set contains longer

[3]Our code and data splits are available at `https://github.com/google-research/language/tree/master/language/nqg`.

examples than the training set. Another is the primitive split, where a given primitive (e.g., "jump") is seen by itself during training, but the test set consists of the primitive recombined with other elements observed during training (e.g., "jump twice"). Other synthetic datasets have been developed to evaluate aspects of compositional generalization beyond SCAN, including NACS (Bastings et al., 2018), CFQ (Keysers et al., 2020), and COGS (Kim and Linzen, 2020).

In addition to introducing the CFQ dataset, Keysers et al. (2020) propose Maximum Compound Divergence (MCD) splits based on the notion of a compound distribution. Their algorithm generates train and test splits that maximize the divergence of their respective compound distributions while bounding the divergence of their respective atom distributions. We extend their methodology to create new TMCD splits for non-synthetic datasets.

Another method for generating train and test splits is the template[4] split (Finegan-Dollak et al., 2018). Unlike the aforementioned evaluations, template splits have been applied to non-synthetic datasets, primarily for text-to-SQL. 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. We analyze and discuss template splits in § 6.1.

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 § 4, all of these models incorporate discrete structure. However, unlike NQG-T5, they have only been evaluated on synthetic parsing tasks.

Recently, 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. Oren et al. (2020) and Zheng and Lapata (2020) also explored compositional generalization on non-synthetic datasets by focusing on the template splits proposed by Finegan-Dollak et al. (2018), demonstrating improvements over standard seq2seq models.

The effect of large-scale pre-training on compositional generalization ability has also been studied. Furrer et al. (2020) finds that pre-training alone cannot solve several compositional generalization challenges, despite its effectiveness across NLP tasks such as question answering (Raffel et al., 2020).

While our work focuses on modeling approaches, compositional data augmentation techniques have also been proposed (Jia and Liang, 2016; Andreas, 2020). NQG-T5 outperforms previously reported results for these methods, but more in-depth analysis is needed.

3 Target Maximum Compound Divergence (TMCD) Splits

The existing evaluations targeting compositional generalization for non-synthetic tasks are template splits and length splits. Here we propose an additional method which expands the set of available evaluations by generating data splits that maximize compound divergence over non-synthetic datasets, termed Target Maximum Compound Divergence (TMCD) splits. As we show in § 6, it results in a generalization problem with different characteristics that can be much more challenging than template splits, and contributes to the comprehensiveness of evaluation.

In standard MCD splits (Keysers et al., 2020), the notion of compounds requires that both source and target 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, example 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 B.3.

Given this definition of compounds, our definition of compound divergence, $\mathcal{D}_C$, is the same as that of Keysers et al. (2020). Specifically,

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

[4]Also referred to as a *query* split.

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. An atom constraint is desirable so that the model knows the possible target atoms to generate. A greedy algorithm similar to the one of Keysers et al. (2020) 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 sets to increase compound divergence without violating the atom constraint, breaking when a swap can no longer be identified.

4 Proposed Approach: NQG-T5

We propose NQG-T5, a hybrid semantic parser that combines a grammar-based approach with a seq2seq model. The two components are motivated by prior work focusing on compositional generalization and natural language variation, respectively, and we show in § 5 that their combination sets a strong baseline for our challenge.

The grammar-based component, **NQG**, consists of a discriminative **N**eural parsing model and a flexible **Q**uasi-synchronous **G**rammar induction algorithm which can operate over arbitrary pairs of strings. Like other grammar-based approaches, NQG can fail to produce an output for certain inputs. As visualized in Figure 3, in cases where NQG fails to produce an output, we return the output from T5 (Raffel et al., 2020), a pre-trained seq2seq model. 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.

We train NQG and T5 separately. Training data for both components consists of pairs of source and target strings, referred to as $\mathbf{x}$ and $\mathbf{y}$, respectively.

4.1 NQG Component

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). 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. 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 $\mathbf{x}$.

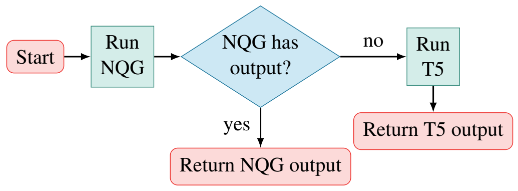

Figure 3: Overview of how predictions are generated by NQG-T5, a simple yet effective combination of T5 (Raffel et al., 2020) with a high-precision grammar-based approach, NQG.

4.1.1 NQG Grammar Induction

Grammar Formalism Synchronous context-free grammars (SCFGs) *synchronously* generate strings in both a source and target language. Compared to related work based on SCFGs for machine translation (Chiang, 2007) 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. Therefore, we adopt the terminology of quasi-synchronous context-free grammars (Smith and Eisner, 2006), or QCFGs, to refer to our induced grammar $\mathcal{G}$.[5] 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, 1965; Younger, 1967) that does not require binarization of the grammar.

Induction Procedure To induce $\mathcal{G}$ from the training data, we propose a QCFG induction algorithm that does not rely on task-specific heuristics or pre-computed word alignments. Notably, our approach makes no explicit assumptions about the source or target languages, beyond those implicit in the QCFG formalism. Table 1 shows examples of induced rules.

Our grammar induction algorithm is guided by the principle of Occam's razor, which leads us to

[5]See Appendix A.1 for additional background on QCFGs.

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
$NT \rightarrow \langle$reviewer, reviewer$\rangle$
$NT \rightarrow \langle$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 induced QCFG rules. The subscript 1 in $NT_{[1]}$ indicates the correspondence between source and target non-terminals.

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 possible derivations.[6] 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:

$$L(\mathcal{R}) = l_N C_N(\mathcal{R}) + l_T C_T(\mathcal{R}) - \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \log_2 \frac{|Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}|}{|Z^{\mathcal{G}}_{\mathbf{x},*}|},$$

where $Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}$ is the set of all derivations in $\mathcal{G}$ that yield the pair of strings $\mathbf{x}$ and $\mathbf{y}$, while $Z^{\mathcal{G}}_{\mathbf{x},*} \supset Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}$ is the set of derivations that yield source string $\mathbf{x}$ and any target string. The constants l_N and l_T can be interpreted as the average bitlength for encoding non-terminal and terminal symbols, respectively. In practice, these are treated as hyperparameters.

We use a greedy search algorithm to find a grammar that approximately minimizes this codelength objective. We initialize $\mathcal{G}$ by creating a rule $NT \rightarrow \langle \mathbf{x}, \mathbf{y} \rangle$ for every training example $(\mathbf{x}, \mathbf{y})$. By construction, the initial grammar perfectly fits the training data, but is also very large. Our algorithm iteratively identifies a rule that can be added to $\mathcal{G}$ that decreases our codelength objective by enabling ≥ 1 rule(s) to be removed, under the invariant constraint that $\mathcal{G}$ can still derive all training examples. The search completes when no rule that decreases the objective can be identified. In practice, we use several approximations to efficiently select a rule at each iteration. Additional details regarding the grammar induction algorithm are described in Appendix A.2.

4.1.2 NQG Semantic Parsing Model

Based on the induced grammar $\mathcal{G}$, we train a discriminative latent variable parsing model, using a method similar to that of Blunsom et al. (2008). We define $p(\mathbf{y} \mid \mathbf{x})$ as:

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

where $Z^{\mathcal{G}}_{\mathbf{x},\mathbf{y}}$ is the set of derivations of $\mathbf{x}$ and $\mathbf{y}$ in $\mathcal{G}$. We define $p(\mathbf{z} \mid \mathbf{x})$ as:

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

where $s(\mathbf{z}, \mathbf{x})$ is a derivation score and the denominator is a global partition function. Similarly to the Neural CRF model of Durrett and Klein (2015), the scores decompose over anchored rules. Unlike Durrett and Klein (2015), we compute these scores based on contextualized representations from a BERT (Devlin et al., 2019) encoder. Additional details regarding the model architecture can be found in Appendix A.3.

At training time, we use a Maximum Marginal Likelihood (MML) objective. We preprocess each example to produce parse forest representations 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. By using dynamic programming to

[6] This can be viewed as a conservative choice, as in practice we expect our neural parser to learn a better model for $P(\mathbf{y}|\mathbf{x})$ than a naive uniform distribution over derivations.

efficiently sum derivation scores inside the training loop, we can efficiently compute the exact MML objective without requiring approximations such as beam search.

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.

4.1.3 NQG Discussion

We note that NQG is 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. Additionally, in contract with 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.

4.2 T5 Component

T5 (Raffel et al., 2020) is a pre-trained sequence-to-sequence Transformer model (Vaswani et al., 2017). We fine-tune T5 for each task.

5 Experiments

We evaluate existing approaches and the newly proposed NQG-T5 across a diverse set of evaluations to assess compositional generalization and handling of natural language variation. We aim to understand how the approaches compare to each other for each type of evaluation and in aggregate, and how the performance of a single approach may vary across different evaluation types.

5.1 Experiments on SCAN and GEOQUERY

For our main experiments, we focus on evaluation across multiple splits of two datasets with compositional queries: SCAN (Lake and Baroni, 2018) and GEOQUERY (Zelle and Mooney, 1996; Tang and Mooney, 2001). The two datasets have been widely used to study compositional generalization and robustness to natural language variation, respectively. Both datasets are closed-domain and have outputs with straightforward syntax, enabling us to make clear comparisons between synthetic vs. non-synthetic setups.

Approaches For NQG-T5, to assess the effect of model size, we compare two sizes of the underlying T5 model: Base (220 million parameters) and 3B (3 billion parameters). To evaluate NQG individually, we treat any example where no output is provided as incorrect when computing accuracy.

We select strong approaches from prior work that have performed well in at least one setting. We group them into two families of approaches described in Figure 1. First, for general-purpose models that have shown strong ability to handle natural language variation, we consider T5, a pre-trained seq2seq model, in both Base and 3B sizes.

Second, for specialized methods with strong compositional biases, we consider approaches that have been developed for SCAN. Some previous approaches for 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 approaches that do not need task-specific resources, we evaluate 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 For the SCAN dataset, 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. (2020).

GEOQUERY (Zelle and Mooney, 1996) contains natural language questions about US geography. Similarly to prior work (Dong and Lapata, 2016, 2018), we replace entity mentions with placeholders. We use a variant of Functional Query Language (FunQL) as the target representation (Kate et al., 2005). In addition to the standard split of Zettlemoyer and Collins (2005), we generate multiple splits focusing on compositional generalization: a new split based on query length and a TMCD split, each consisting of 440 train and 440 test examples. We also generate a new template split consisting of 441 train and 439 test examples.[7]

[7]We generate a new template split rather than use the GEOQUERY template split of Finegan-Dollak et al. (2018) to avoid overlapping templates between the train and test sets when mapping from SQL to FunQL.

	SCAN				GEOQUERY				Avg.
System	Jump	Turn Left	Len.	MCD	Standard	Template	Len.	TMCD	Rank
LANE (Liu et al., 2020)	**100**	—	**100**	**100**	—	—	—	—	—
NSSM (Chen et al., 2020)	**100**	—	**100**	—	—	—	—	—	—
Syntactic Attn. (Russin et al., 2019)	91.0	**99.9**	15.2	2.9	77.5	70.6	23.6	0.0	3.9
CGPS (Li et al., 2019)	98.8	**99.7**	20.3	2.0	62.1	32.8	9.3	32.3	4.4
GECA (Andreas, 2020)	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., 2020)	**99.5**	62.0	14.4	15.4	**92.9**	87.0	39.1	54.3	2.9
T5-3B (Raffel et al., 2020)	**99.0**	65.1	3.3	11.6	**93.2**	83.1	36.8	51.6	—
NQG-T5-Base	**100**	**100**	**100**	**100**	**92.9**	**88.8**	**52.2**	**56.6**	**1.0**
NQG-T5-3B	**100**	**100**	**100**	**100**	**93.7**	85.0	**51.4**	54.1	—
NQG	**100**	**100**	**100**	**100**	76.8	61.9	37.4	41.1	2.3

Table 2: **Main Results.** Existing approaches do not excel on a diverse set of evaluations across synthetic and non-synthetic tasks, but NQG-T5 obtains significant improvements. For comparison, we report the average rank among 5 approaches across all 8 evaluations. Gray cells are previously reported results. † indicates differences in GEOQUERY settings (see discussion in § 5.1). Boldfaced results are within 1.0 points of the best result.

We report exact-match accuracy for both datasets.[8] Hyperparameters and pre-processing details can be found in Appendix B.

Results The results are presented in Table 2. The results for T5 on SCAN are from Furrer et al. (2020). Additionally, we include results for GECA[9] (Andreas, 2020), a data augmentation method, as well as LANE (Liu et al., 2020) and NSSM (Chen et al., 2020)[10]. We also compare with SpanBasedSP[11] (Herzig and Berant, 2020).

From the results, we first note that the relative performance of approaches on compositional splits of SCAN is not very predictive of their relative performance on compositional splits of GEOQUERY. For example, GGPS is better than T5 on the length split of SCAN but is significantly worse than T5 on the length split of GEOQUERY. Similarly, the ranking of most methods is different on the (T)MCD splits of the two datasets. Second, the proposed NQG-T5 approach combines the strengths of T5 and NQG to achieve superior results across all evaluations. It improves over T5 on compositional generalization for both synthetic and non-synthetic data while maintaining T5's performance on handling in-distribution natural language variation, leading to an average rank of 1.0 compared to 2.9 for T5. (To the best of our knowledge, both T5 and NQG-T5 achieve new state-of-the-art accuracy on the standard split of GEOQUERY.)

Finally, we note that there is substantial room for improvement on handling both compositional generalization and natural language variation.

5.2 Experiments on SPIDER

We now compare the approaches on SPIDER (Yu et al., 2018), a non-synthetic text-to-SQL dataset that includes the further challenges of schema linking and modeling complex SQL syntax.

SPIDER contains 10,181 questions and 5,693 unique SQL queries across 138 domains. The primary evaluation is in the cross-database setting, where models are evaluated on examples for databases not seen during training. The primary challenge in this setting is generalization to new database schemas, which is not our focus. Therefore, we use a setting where the databases are shared between train and test examples.[12] We gen-

[8]For GEOQUERY we report the mean of 3 runs for NQG, with standard deviations reported in Appendix B.5

[9]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.

[10]These SCAN-motivated approaches both include aspects of discrete search and curriculum learning, and have not been demonstrated to scale effectively to non-synthetic parsing tasks. Moreover, the code is either not yet released (NSSM) or specialized to SCAN (LANE).

[11]SpanBasedSP preprocesses SCAN to add program-level supervision. For GEOQUERY, they similarly use FunQL, but uses slightly different data preprocessing and report denotation accuracy. We computed NQG-T5's denotation accuracy to be 2.1 points *higher* than exact-match accuracy on the standard split of GeoQuery.

[12]This is similar to the "example split" discussed in Yu et al. (2018). However, we only consider examples in the original training set for databases with more than 50 examples to ensure sufficient coverage over table and column names in

System	Spider-SSP Rand.	Templ.	Len.	TMCD
T5-Base −*schema*	76.5	45.3	42.5	42.3
T5-Base	82.0	59.3	49.0	60.9
T5-3B	85.6	64.8	56.7	69.6
NQG-T5-Base	81.8	59.2	49.0	60.8
NQG-T5-3B	85.4	64.7	56.7	69.5
NQG	1.3	0.5	0.0	0.5

Table 3: Results on Spider-SSP. While the text-to-SQL task is not modeled well by the NQG grammar due to SQL's complex syntax, NQG-T5 still performs well by relying on T5.

System	Spider-XSP Dev
RYANSQL v2 (Choi et al., 2020)	70.6
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 and NQG-T5 are competitive with the state-of-the-art.

erate 3 new splits consisting of 3,282 train and 1,094 test examples each: a random split, a split based on source length, and a TMCD split. We also generate a template split by anonymizing integers and quoted strings, consisting of 3,280 train and 1,096 test examples. 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 similarly 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. 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 on handling unseen schema rather than compositional generalization. To our surprise, T5-3B proves to be competitive with the state-of-the-art (Choi et al., 2020) for approaches without access to database contents beyond the table and column names. As NQG-T5 simply uses T5's output when the induced grammar lacks coverage, 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. For GeoQuery, the TMCD split is significantly more challenging than the template split. However, for Spider, the template and TMCD splits are similarly challenging. Notably, template splits do not have an explicit atom constraint. We find that for the Spider template split, T5-Base accuracy is 53.9% for the 30.3% of test set examples that contain an atom not seen during training, and 61.6% on the remainder, indicating that generalization to unseen atoms can contribute to the difficulty of template splits.[13] Length splits are also very challenging, but they lead to a more predictable error pattern for seq2seq models, as discussed next.

6.2 T5 Analysis

We analyze NQG-T5's components, starting with T5. On length splits, there is a consistent pattern to the errors. T5's outputs on the test set are not significantly longer than the maximum length observed during training, leading to poor performance. This phenomenon was explored by Newman et al. (2020).

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. Non-exclusively with these errors, 53% of the incorrect predictions occur when the gold target contains an atom that is seen in only 1

the training data. This includes 51 databases.

[13]Future work could explore different choices for constructing template and TMCD splits, such as alternative compound definitions and atom constraints.

	SCAN				GEOQUERY				SPIDER-SSP			
Metric	Jump	Turn L.	Len.	MCD	Stand.	Templ.	Len.	TMCD	Rand.	Templ.	Len.	TMCD
NQG Coverage	100	100	100	100	80.2	64.5	43.3	43.7	1.5	0.5	0.0	0.6
NQG Precision	100	100	100	100	95.7	95.8	86.4	94.1	87.5	83.3	—	85.7

Table 5: NQG coverage and precision. NQG-T5 outperforms T5 when NQG has higher precision than T5 over the subset of examples it covers.

Dataset	Split	$\%_{ZA}$	$\mathcal{D}_C$	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	Template	30.3	0.08	59.2
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 ($\%_{ZA}$), compound divergence ($\mathcal{D}_C$), and T5-Base accuracy for various dataset splits.

example during training, suggesting that T5 struggles with single-shot learning of new atoms. In other cases, the errors appear to reflect over-fitting to spurious correlations between inputs and outputs. Some error examples are shown in Appendix B.6.

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 coverage on some data splits.

There is a significant difference in the effectiveness of the grammar induction procedure among the three datasets. Induction is particularly unsuccessful for SPIDER, as SQL has complicated syntax and often requires complex coordination across discontinuous clauses. 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 representing nested sub-structures. The degree of span-to-span correspondence between natural language and SQL is seemingly lower than for other formalisms such as FunQL, which limits the effectiveness of grammar induction. Intermediate representations for SQL such as SemQL (Guo et al., 2019) may help increase the correspondence between source and target syntax.

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 directly represent relations between source strings, such as semantic similarity, or relations between target strings, such as logical equivalence (e.g. `intersect(a,b)` ⇔ `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 experiments and analysis demonstrate that NQG and T5 offer different strengths. NQG generally has higher precision for out-of-distribution examples, but is limited by the syntactic constraints of the grammar formalism and by requiring exact lexical overlap with induced rules in order to provide a derivation at inference time. T5's coverage is not limited by such constraints, but precision can be significantly lower for out-of-distribution examples. With NQG-T5, we offer a simple combination of these strengths. While accuracy is still limited for out-of-distribution examples where NQG lacks coverage, we believe it sets a strong and simple baseline for future work.

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.

Acknowledgements

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

Ethical Considerations

This paper proposed to expand the set of benchmarks used to evaluate compositional generalization in semantic parsing. While we hope that ensuring semantic parsing approaches perform well across a diverse set of evaluations, including ones that test out-of-distribution compositional generalization, would lead to systems that generalize better to languages not well represented in small training sets, we have only evaluated our methods on semantic parsing datasets in English.

Our NQG-T5 method uses a pre-trained T5 model, which is computationally expensive in finetuning and inference, especially for larger models (see Appendix B.1 for details on running time and compute architecture). Our method does not require pre-training of large models, as it uses pre-existing model releases. NQG-T5-base outperforms or is comparable in accuracy to T5-3B on the non-SQL datasets, leading to relative savings of computational resources.

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

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. *Advances in Neural Information Processing Systems*, 33.

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

DongHyun Choi, Myeongcheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. RYANSQL: recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *arXiv preprint arXiv:2004.03125*.

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. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

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. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

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.

T. Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

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, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR*.

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

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.

B. M. Lake, T. Linzen, and M. Baroni. 2019. Human few-shot learning of compositional instructions. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society*.

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

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

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. 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 4284–4293, Hong Kong, China. Association for Computational Linguistics.

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. *Advances in Neural Information Processing Systems*, 33.

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. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 276–291, Online. Association for Computational Linguistics.

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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2482–2495.

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. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

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.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer.

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.

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, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 2*, pages 1050–1055.

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.

Hao Zheng and Mirella Lapata. 2020. Compositional generalization via semantic tagging. *arXiv preprint arXiv:2010.11818*.

Appendix

We organize the appendix into two sections:

- Additional details for NQG in Appendix A.
- Additional experimental details and analysis in Appendix B.

A NQG Details

In this section we describe the NQG grammar induction algorithm and parsing model in detail, starting with relevant background and notation for QCFGs.

A.1 Background: SCFGs and QCFGs

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. An example derivation is shown in Figure 4.

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 $\Rightarrow$.

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).

Compositionality Notably, grammar formalisms such SCFGs and QCFGs capture the formal notion of the principle of compositionality as a homomorphism between source and target structures (Montague, 1970; Janssen and Partee, 1997).

A.2 NQG Grammar Induction Details

Having defined the codelength scoring function that we use to compare grammars in section 4.1.1, we describe our greedy search algorithm that finds a grammar that approximately minimizes this objective.[14]

Initialization 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 $\mathbf{x}, \mathbf{y} \in \mathcal{D}$.[15]

Optimization Algorithm Our 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. At a high level, our greedy algorithm iteratively identifies a rule to be added to $\mathcal{R}$ that decreases the codelength 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 every $\mathbf{x}, \mathbf{y} \in \mathcal{D}$. The search completes when no rule that decreases $L(\mathcal{R})$ can be identified.

To describe the implementation, 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, $\mathcal{R}$:

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

We define an operation SPLIT that generates possible choices for *splitting* a rule into 2 rules:

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

[14]The induction objective contains hyperparameters representing the bitlength of terminal and non-terminal symbols. For all experiments we use $l_N = 1$. For GEOQUERY and SPIDER we use $l_T = 8$, and use $l_T = 32$ for SCAN.

[15]These initialization rules are used for GEOQUERY and SPIDER, but SCAN does not contain any exact token overlap between source and target languages.

$NT \rightarrow \langle$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$\rangle$

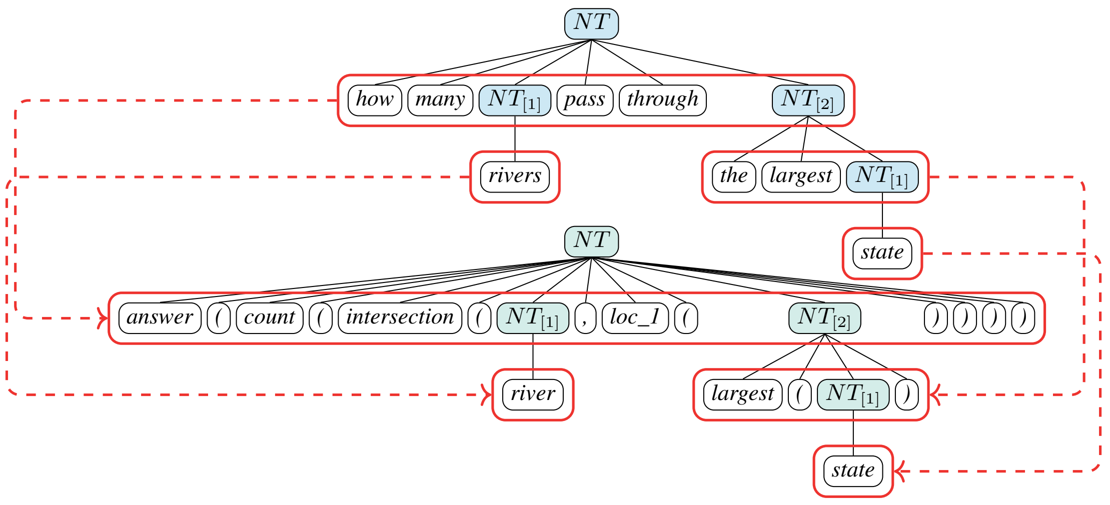

Figure 4: An example QCFG derivation. Each non-terminal in the source derivation (blue) corresponds to a non-terminal in the target derivation (green). The QCFG rules used in the derivation are shown above.

where g and h 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.[16]

SPLIT can be implemented by considering pairs of sub-strings in 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 $NT_{[1]}$, largest ($NT_{[1]}$)$\rangle$" and "$NT \rightarrow \langle$state, state$\rangle$". This step can require re-indexing of non-terminals.

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 NEW that returns a set of candidate rules to consider:

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

Similarly, we can compute the set of rules that are made redundant and can be eliminated by introducing one these candidate rules, f:

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

[16]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.

We can then define the codelength reduction of adding a particular rule, $-\Delta L(\mathcal{R}, f) = L(\mathcal{R}) - L(\mathcal{R}')$ where $\mathcal{R}' = (\mathcal{R} \cup f) \setminus \text{ELIM}(\mathcal{R}, f)$.[17] Finally, we can select the rule with the largest $-\Delta L$:

$$\text{MAX}(\mathcal{R}) = \underset{f \in \text{NEW}(\mathcal{R})}{\text{argmax}} -\Delta L(\mathcal{R}, f)$$

Conceptually, after initialization, the algorithm then proceeds as:

```
while $|\text{NEW}(\mathcal{R})| > 0$ do
    $r \leftarrow \text{MAX}(\mathcal{R})$
    if $-\Delta L(\mathcal{R}, r) < 0$ 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

[17]The last term of the codelength objective described in section 4.1.1 is related to the increase in the proportion of incorrect derivations due to introducing f. 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 estimating the increase in incorrect derivations over this sample only. We sample $k = 10$ examples for all experiments.

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.[18]

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.

Our grammar induction algorithm is similar to the 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; Tang and Mooney, 2001; Ravi and Knight, 2009; Poon et al., 2009).

A.3 NQG Parsing Model Details

In this section we provide details on how we generate derivation scores, $s(\mathbf{z}, \mathbf{x})$, using a neural model, as introduced in § 4.1. The derivation 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., 2019) 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. Our formulation for encoding spans is similar to that used in other neural span-factored models (Stern et al., 2017; Lee et al., 2017).

B Experimental Details

B.1 Model Hyperparameters and Runtime

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 a given dataset.

For T5, we selected a learning rate of $1e^{-4}$ from $[1e^{-3}, 1e^{-4}, 1e^{-5}]$, which we used for all experiments. Otherwise, we used the default hyperparameters for fine-tuning. We fine-tune for $3,000$ steps for GEOQUERY and $10,000$ for SPIDER. T5-Base trained with a learning rate of $1e^{-4}$ reached 94.2% accuracy at $3,000$ steps on a random split of the standard GeoQuery training set into 500 training and 100 validation examples.

For the NQG neural model, we use the pre-trained BERT Tiny model of Turc et al. (2019) (4.4M parameters) for SCAN and SPIDER, and BERT Base (110.1M parameters) for GEOQUERY, where there is more headroom for improved scoring. We do not freeze pre-trained BERT parameters during training. For all experiments, we use $d = 256$ dimensions for computing anchored rule scores. We fine-tune for 256 steps and use a learning rate of $1e^{-4}$. We use a batch size of 256.

We train NQG on 8 V100 GPUs. Training NQG takes < 5 minutes for SCAN and SPIDER (BERT Tiny), and up to 90 minutes for GEOQUERY (BERT Base). We fine-tune T5 on 32 Cloud TPU v3 cores.[19] For GEOQUERY, fine-tuning T5 takes approximately 5 and 37 hours for Base and 3B, respectively. For SPIDER, fine-tuning T5 takes approximately 5 and 77 hours for Base and 3B, respectively.

B.2 Dataset Preprocessing

For 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.

For 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

[18]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.

[19]https://cloud.google.com/tpu/

Source: how many states are next to major rivers
Target: `answer ( count ( intersection ( state , next_to_2 ( intersection ( major , river ) ) ) ) )`
Prediction: `answer ( count ( intersection ( state , next_to_2 ( intersection ( major , intersection ( river , m0 ) ) ) ) ) )`
Notes: The trigram "`major , intersection`" occurs 28 times during training, but "`major , river`" occurs 0 times. In this case, T5 also hallucinates "`m0`" despite no entity placeholder occuring the source.

Source: which state has the highest peak in the country
Target: `answer ( intersection ( state , loc_1 ( highest ( place ) ) ) )`
Prediction: `answer ( highest ( intersection ( state , loc_2 ( highest ( intersection ( mountain , loc_2 ( m0 ) ) ) ) ) ) )`
Notes: The token "`highest`" occurs after "`answer (`" in 83% of instances in which "`highest`" occurs in the training set. Note that T5 also hallucinates "`m0`" in this case.

Table 7: Example prediction errors for T5-Base for the GEOQUERY TMCD split.

Dataset	Examples	Induced Rules	Ratio
SCAN	16727	21	796.5
GEOQUERY	600	234	2.6
SPIDER-SSP	3282	4155	0.79

Table 8: Sizes of induced grammars.

	Std.	Templ.	Len.	TMCD
NQG-T5-3B Acc.	0.6	1.2	1.2	0.4
NQG-T5-Base Acc.	0.5	1.4	1.1	0.4
NQG Acc.	1.2	4.5	1.5	0.4
NQG Coverage	0.7	3.4	1.8	0.1
NQG Precision	0.7	1.9	1.7	1.2

Table 9: Standard deviation of NQG for GEOQUERY.

source sequence length of 512 for T5, this leads to some schema strings being truncated (affecting about 5% of training examples).

SCAN did not require any dataset-specific preprocessing.

B.3 Atom and Compound Definitions

For GEOQUERY, the tree structure of FunQL 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(_), _)`.

For SPIDER, 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.

B.4 Grammar Sizes

Induced grammar sizes for a selected split of each dataset are shown in Table 8. For SPIDER, the number of induced rules is larger than the original dataset due to the identity rules added during initialization.

B.5 GEOQUERY Variance

In tables 2 and 5 we report the mean of 3 runs for NQG for GEOQUERY. The standard deviations for these runs are reported in Table 9. The reported standard deviations for NQG-T5 use the same fine-tuned T5 checkpoint, so they do not reflect any additional variance from different fine-tuned T5 checkpoints.

B.6 T5 GEOQUERY Errors

We include several example T5-Base errors on the GEOQUERY TMCD split in Table 7.