# *Finding needles in a haystack*: Sampling Structurally-diverse Training Sets from Synthetic Data for Compositional Generalization

**Inbar Oren**[1] **Jonathan Herzig**[1] **Jonathan Berant**[1,2]
[1]School of Computer Science, Tel-Aviv University
[2]Allen Institute for Artificial Intelligence
inbaroren@mail.tau.ac.il, {jonathan.herzig,joberant}@cs.tau.ac.il

## Abstract

Modern semantic parsers suffer from two principal limitations. First, training requires expensive collection of utterance-program pairs. Second, semantic parsers fail to generalize at test time to new compositions/structures that have not been observed during training. Recent research has shown that automatic generation of synthetic utterance-program pairs can alleviate the first problem, but its potential for the second has thus far been under-explored. In this work, we investigate automatic generation of synthetic utterance-program pairs for improving compositional generalization in semantic parsing. Given a small training set of *annotated* examples and an "infinite" pool of *synthetic* examples, we select a subset of synthetic examples that are structurally-diverse and use them to improve compositional generalization. We evaluate our approach on a new split of the schema2QA dataset, and show that it leads to dramatic improvements in compositional generalization as well as moderate improvements in the traditional i.i.d setup. Moreover, structurally-diverse sampling achieves these improvements with as few as 5K examples, compared to 1M examples when sampling uniformly at random – a 200x improvement in data efficiency.

## 1 Introduction

Semantic parsers map natural language utterances to executable programs (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005). A worrying weakness of semantic parsers that has been recently exposed, is their inability to generalize at test time to new compositions (Finegan-Dollak et al., 2018; Lake and Baroni, 2018; Keysers et al., 2020; Kim and Linzen, 2020; Gu et al., 2021). For example, a virtual assistant trained on the examples *"Show me Thai restaurants that allow pets"* and *"How many hotels are there in Tokyo"*, might not generalize to *"How many hotels in Tokyo allow pets?"*. This type of out-of-domain generalization to new compositions constructed from components seen during training is commonly termed *compositional generalization*.

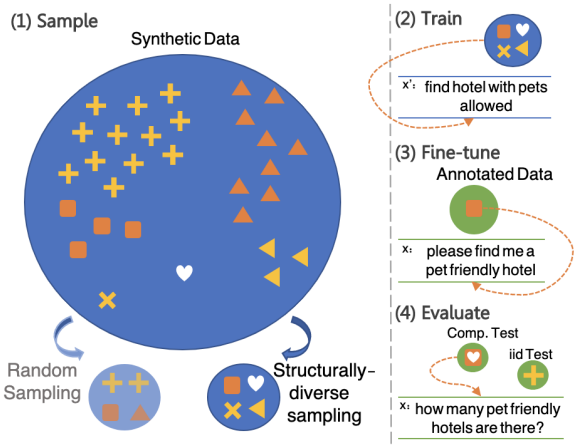


Figure 1: Given a small set of annotated data and a large pool of synthetic examples, we propose to sample a diverse training set of synthetic examples which includes a myriad of structures. Training a semantic parser on the synthetic data and fine-tuning on annotated data dramatically improves the parser's compositional generalization.

Two high-level approaches have been considered for tackling compositional generalization: (a) designing models with a stronger compositional inductive bias (Liu et al., 2020; Russin et al., 2020; Zheng and Lapata, 2020; Herzig and Berant, 2021), and (b) adding training data that will encourage a compositional solution (Akyürek et al., 2021; Guo et al., 2020c; Wang et al., 2021; Guo et al., 2020a). In the latter approach, typically a model is trained from labeled data (Jia and Liang, 2016; Yu et al., 2020; Zhong et al., 2020), and is used to later generate new examples. An alternative approach is to use a manually-built synchronous grammar that automatically generates programs paired with synthetic utterances (Wang et al., 2015; Cheng et al., 2018; Weir et al., 2020). Using a grammar allows generating large amounts of synthetic data that cover a



*Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10793–10809
November 7–11, 2021. 

wide range of program structures. This has been shown to be useful in the i.i.d setup, and combined with paraphrase models, has led to high-accuracy parsers that are trained from synthetic data *only* (Xu et al., 2020b).

In this work, we investigate the potential of using synthetic data, generated from a synchronous grammar, to improve compositional generalization. Tsarkov et al. (2020) have shown that large quantities of such data improve compositional generalization. However, they evaluated on synthetic utterances only, and did not examine generalization to natural language. Moreover, error rates were high in some compositional splits even when the training set was as large as 1M examples. In this work, we ask whether we can strategically sample a small and structurally-diverse training set and improve compositional generalization without incurring a high cost to training and consequently, to the environment (Schwartz et al., 2020). We hypothesize that a training set that encompasses a diverse set of structures can steer the model towards a compositional solution.

We examine a realistic setup, where we have a small labeled training set (∼1,000 examples) and a large pool of synthetic utterances paired with programs (Fig. 1), which are queries over a database (DB) in our setup. We propose two methods for strategically sampling a diverse synthetic training set. In the first, termed *uniform abstract template (UAT)* sampling, we abstract queries by replacing DB constants with abstract tokens (e.g., replacing `petsAllowed` with `property`), and then skew the distribution towards uniform sampling over the derived templates. This increases structure diversity in the training set, which intuitively should lead to better compositional generalization. In the second method, termed *compound maximum entropy (CMaxEnt)* sampling, we consider the tree structure of every DB query, and following Tsarkov et al. (2020) define *compounds*, which are sub-trees in the query. We then heuristically solve an optimization problem, where our goal is to select a training set that has maximal entropy over compounds. This results in a training set with a diverse set of substructures, which should enhance compositional generalization.

We evaluate our approach on a new split of the Schema2QA dataset (Xu et al., 2020a), in which it has been shown that synthetic data can lead to high accuracy parsers in the i.i.d setup (Xu et al., 2020b). We train an encoder-decoder model on synthetic data and subsequently fine-tune it on the small annotated data. We show that random sampling of synthetic data improves performance, where gradually increasing the size of the synthetic training set also improves compositional generalization. With 1M randomly-sampled examples, accuracy in the compositional split improves from 20.1→37.7 and in the i.i.d split from 81.2→85.0. When sampling structurally-diverse data, compositional generalization improves from 20.1 to >40 with as few as 5K examples, outperforming training with 1M synthetic examples. In addition, the i.i.d generalization is comparable to random sampling. UAT and CMAXENT both lead to large improvements in compositional generalization, but UAT is more effective than CMAXENT.

Overall, our work demonstrates that sampling diverse structures from synthetic data can lead to dramatic gains in compositional generalization at negligible cost, while preserving or improving performance in the i.i.d setup. Our code and data can be downloaded from `http://github.com/inbaroren/scfg-sampling-for-comp-gen`.

## 2 Problem Setup

We assume access to a small dataset of *natural language* utterances paired with queries, $\mathcal{D}_{train} = \{(x_i, z_i)\}_{i=1}^{N_{NL}}$, and a large pool of *synthetic* utterances paired with queries $\mathcal{D}_{train}^{Syn} = \{(x'_i, z_i)\}_{i=1}^{N_{Syn}}$, where $N_{NL} \ll N_{Syn}$. In this work, $\mathcal{D}_{train}^{Syn}$ is generated with a synchronous context-free grammar, which provides wide coverage of query structures and tight control over the generated queries, but other methods of generating synthetic examples are possible (Andreas, 2020; Guo et al., 2020a; Wang et al., 2021). Table 1 provides examples of natural language utterances, synthetic utterances, and queries in the ThingTalk language, a language designed for virtual assistants used in this work (through the Schema2QA dataset (Xu et al., 2020a,b)).

Our goal is to train a model using $\mathcal{D}_{train}$ and $\mathcal{D}_{train}^{Syn}$ and generalize to a test set $\mathcal{D}_{test}$ sampled from the same distribution as $\mathcal{D}_{train}$. More importantly, our model should generalize to a *compositional test set*, $\mathcal{D}_{test}^{Comp}$, which contains structures/compositions that are not observed in $\mathcal{D}_{train}$ or $\mathcal{D}_{train}^{Syn}$. We now describe this test set.

| | |
|---|---|
| $x$: | *show me a book with at least 2 awards .* |
| $x'$: | *which books have more than 2 awards* |
| $z$: | `( @Book ) filter count ( award:Array(String) ) >= 2` |
| $x$: | *search for any books with a rating of 5 that also have 100 pages or more* |
| $x'$: | *what book gets number of pages at least 100 and gets the 5 mark ?* |
| $z$: | `( @Book ) filter ratingValue:Number == 5 and numberOfPages:Number >= 100` |
| $x$: | *show me hotels with a fitness center* |
| $x'$: | *is there any hotels having fitness center in its amenity features* |
| $z$: | `( @Hotel ) filter amenityFeature:Array(LocationFeatureSpecification) contains "fitness center"` |
| $x$: | *can you find a hotel that accepts dogs ?* |
| $x'$: | *what hotels having pets allowed ?* |
| $z$: | `( @Hotel ) filter petsAllowed: Boolean == true` |
| $z_{abs}$: | `( @table ) filter property: type op entity` |

Table 1: Examples from Schema2QA of utterance-query pairs ($x$,$z$) with their synthetic utterances ($x'$), in the books and hotels domains. In the bottom example, the abstract template ($z_{abs}$) is included. Queries are in abbreviated syntax for clarity.

**Compositional split** We define a compositional split following the *abstract template split* proposed by Finegan-Dollak et al. (2018), which is commonly used for assessing compositional generalization (Lee et al., 2019; Andreas, 2020; Oren et al., 2020). In this approach, queries are abstracted into templates that correspond to different structures. Templates are then partitioned into disjoint sets (train/development/test), which ensures that test time structures are never observed at training time. While prior work focused on abstracting entities only, by replacing any DB entity with the token `entity`, in this work we abstract queries into more coarse templates, e.g, table constants are replaced by the token `table`. Table 2 lists all abstracted query parts and their corresponding abstract token, and Table 1 (bottom) shows an example query $z$ and its abstract template $z_{abs}$. Splitting with coarse templates increases the distance between train and test structures, which in turn increases the need for compositionality.[1] Table 3 shows four examples, where the first two examples and the last two share an abstract template. In an *i.i.d split* the first two (and last two) examples can appear in different sets, but in a *compositional split* they must be either in the training set or test set, requiring compositional generalization.

**Research questions** Our experimental setup, where $\mathcal{D}_{test}^{Comp}$ contains structures that are not observed in $\mathcal{D}_{train}$ or $\mathcal{D}_{train}^{Syn}$, allows us investigate several questions. First, does training on the large synthetic dataset $\mathcal{D}_{train}^{Syn}$ improve generalization to $\mathcal{D}_{test}^{Comp}$ compared to training on $\mathcal{D}_{train}$ only? Second, if $\mathcal{D}_{train}^{Syn}$ improves compositional generalization, can we make it more sample-efficient? Specifically, can we sample a smaller set $\hat{\mathcal{D}}_{train}^{Syn}$, such that $|\hat{\mathcal{D}}_{train}^{Syn}| \ll |\mathcal{D}_{train}^{Syn}|$ and still improve compositional generalization. Last, can we do the above while preserving or improving generalization to the i.i.d test set, $\mathcal{D}_{test}$? Answering these questions will be the focus of §3 and §4.

| Category | Token | Example |
|---|---|---|
| Entity | `entity` | `true`, `"Tokyo"` |
| Table | `@table` | `@org.schema.Book.Book` |
| Table property | `property` | `petsAllowed`,`numberOfPages` |
| Entity or property type | `type` | `Array(String)`, `Number` |
| Operator | `op` | `>=`, `and`, `not` |
| Function | `func` | `count`, `sum` |
| Modifier | `func_mod` | `asc`, `desc` |

Table 2: List of all parts of the query we abstract and their corresponding abstract token.

## 3 Sampling a Structurally-diverse Training Set

We first succinctly describe our model and training procedure (§3.1) and then turn to methods for sampling structurally-diverse training sets.

### 3.1 Model and Training

In this work, we start from a pre-trained encoder-decoder model (Lewis et al., 2020), as such models have been shown to provide a good initialization for fine-tuning semantic parsers (Furrer et al., 2020; Herzig et al., 2021). We then train our model in two steps (Yu et al., 2020; Wang et al., 2021). First, on synthetic utterance-query pairs $(x', z) \in \hat{\mathcal{D}}_{train}^{Syn}$,

[1] Preliminary experiments on synthetic data pointed that abstracting entities only might not lead to a compositional split that is challenging enough.

| | Example | iid split | comp. split |
|---|---|---|---|
| $x'$: | *please search the hotels with pets allowed* | train | train |
| $z$: | `( @Hotel ) filter patsAllowed:Boolean == true` | | |
| $z_{abs}$: | `( @table ) filter property:type op entity` | | |
| $x'$: | *please search books with ebook format* | test | train |
| $z$: | `( @Book ) filter format:Enum == ebook` | | |
| $z_{abs}$: | `( @table ) filter property:type op entity` | | |
| $x'$: | *how many people are there* | train | test |
| $z$: | `aggregate count of ( @Person )` | | |
| $z_{abs}$: | `func ( @table )` | | |
| $x'$: | *how many hotels are there* | test | test |
| $z$: | `aggregate count of ( @Hotel )` | | |
| $z_{abs}$: | `func ( @table )` | | |

Table 3: A compositional split prohibits the same abstract template to appear in both the training and test set, and hence tests compositional generalization. Above, examples 1-2 and 3-4 share the same template, so in an i.i.d split they can be assigned to different sets, while in a compositional split they must be either in the training or test set.

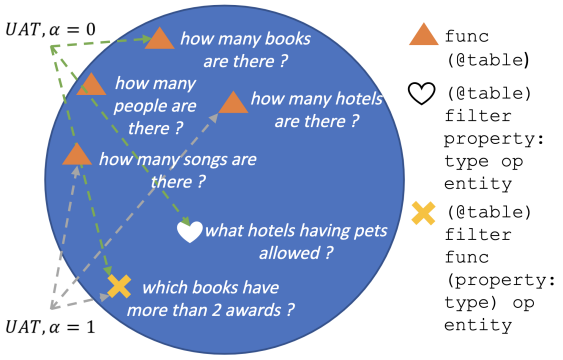


Figure 2: UAT sampling. When $\alpha = 1$, UAT is a uniform sample over $\mathcal{D}^{Syn}_{train}$. As $\alpha \to 0$, the probability over abstract templates ($z_{abs}$) becomes uniform. Consequently, the sample includes more abstract templates.

and then on natural langauge utterance-query pairs $(x, z) \in \mathcal{D}_{train}$. Training in two steps mitigates the gap in language variation between $\hat{\mathcal{D}}^{Syn}_{train}$ and $\mathcal{D}_{train}$. We train with the standard maximum-likelihood objective, maximizing the probability of the gold sequence, $z$.

**Uniform sampling** Our baseline sampling method is to construct $\hat{\mathcal{D}}^{Syn}_{train}$ by sampling from $\mathcal{D}^{Syn}_{train}$ uniformly. This simulates sampling from the synchronous grammar directly, and can inform us whether synthetic data improves generalization to $\mathcal{D}^{Comp}_{test}$, even if $\hat{\mathcal{D}}^{Syn}_{train}$ is very large.

### 3.2 Uniform Abstract Template Sampling

We conjecture that a model is more likely to converge to a "compositional solution" if it observes at training time a multitude of different structures, and learns that sub-structures can occur in multiple contexts. For example, if two properties always co-occur in the training set (e.g., `ratingValue` and `numberOfPages` in the second example of Table 1), then the model might erroneously learn to always decode them together.

To achieve this goal, we define a sampling process, UAT, that results in a more uniform distribution over abstract templates. In this process, we first sample an abstract template, and then sample an example conditioned on that template. We skew the distribution over templates to be close to uniform, which leads to templates that have few examples to be over-represented in the training set $\hat{\mathcal{D}}^{Syn}_{train}$. Thus, minimizing the loss over the training set will take into account a large number of templates, which should improve compositional generalization. Typically, even if the number of examples in $\mathcal{D}^{Syn}_{train}$ is large (∼6M in our experiments), the number of abstract templates is much smaller (251 in our experiments, see §4). Fig. 2 illustrates this sampling process.

Formally, we construct $\hat{\mathcal{D}}^{Syn}_{train}$ by sampling from $\mathcal{D}^{Syn}_{train}$ without replacement using the following procedure. Let $\mathcal{T}$ be the set of templates in $\mathcal{D}^{Syn}_{train}$, $T(z_i)$ be the abstract template for a query $z_i$, and $c(T(z_i))$ be the number of times $T(z_i)$ occurs in $\mathcal{D}^{Syn}_{train}$. We estimate a probability distribution over templates, $p(T(z_i)) = \frac{c(T(z_i))}{|\mathcal{D}^{Syn}_{train}|}$, and a distribution over examples conditioned on a particular template $u_{T(z_i)}((x'_i, z_i)) = \frac{1}{c(T(z_i))}$. Now we sample a synthetic utterance-query pair from the following distribution:

$$q_\alpha((x'_i, z_i)) = \frac{p(T(z_i))^\alpha}{\sum_{T \in \mathcal{T}} p(T)^\alpha} \cdot u_{T(z_i)}((x'_i, z_i)), \quad (1)$$

where $\alpha \in [0, 1]$. When $\alpha = 1$, this corresponds to the aforementioned uniform sampling over examples (factorized over templates), but when $\alpha = 0$, this corresponds to uniform sampling over the templates for which there still remain examples to be sampled. Values between 0 and 1 allow a smooth transition between uniform sampling over examples and over templates. In §4, we will examine the effect of various values of $\alpha$ on compositional generalization for varying sizes of the sampled training set, $\hat{\mathcal{D}}^{Syn}_{train}$.

### 3.3 Compound Maximum Entropy

UAT sampling does not consider the similarity between different abstract templates, treating each template independently. However, different tem-

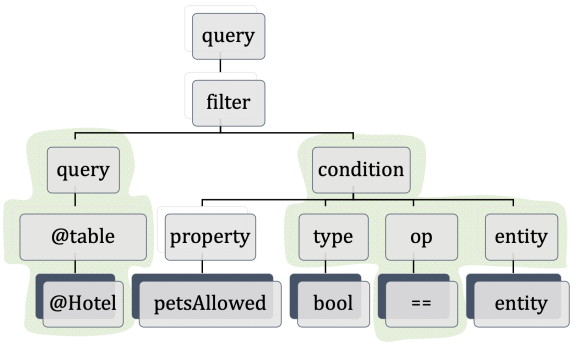


Figure 3: A ThingTalk parse tree (with abstract entities). Each node is an *atom*, and any subgraph of height at most 2 that has at least one terminal is a *compound*. Two compounds are marked in green.

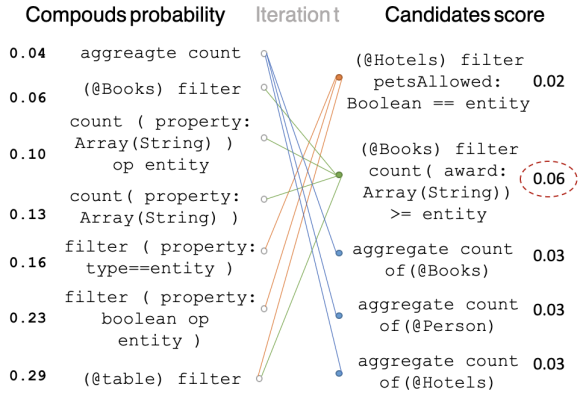


Figure 4: One iteration in the CMAXENT sampling. On the left is a set of compounds and their probability $p(c)$ in the current training set. on the right is a list of candidate queries and the gain in compound entropy if they are selected. Colored lines show what compounds appear in what queries. Here, the second query is selected, and added to the sample with one of its accompanying utterances.

plates potentially share some sub-structure that can be used for obtaining more diversity at the *sub-structure* level. We now consider CMAXENT, a method for sampling a synthetic training set with diverse sub-structures.

Recently, Keysers et al. (2020) and Shaw et al. (2020) used the notion of sub-structures, to construct a compositional test set. Given a program tree, they define *atoms* to be nodes in the tree and *compounds* to be sub-structures in the tree. Then, given a pool of examples, they partition it into two sets (train and test), such that the distribution over atoms is similar, but the distribution over compounds is different. Here, we adopt their definition of atoms and compounds, but for a different objective. We aim to sample a set $\hat{\mathcal{D}}^{Syn}_{train}$, such that the entropy over atoms and compounds is maximized. This will expose the model to a diverse set of atoms and compounds in multiple contexts, which should lead to compositional generalization.

**Atom and compound distributions** Queries in formal languages can be parsed into trees. We adopt the definition of Keysers et al. (2020), and define *atoms* as any node in the tree, and *compounds* as any tree of height $\leq 2$, that includes at least one tree terminal. We reduce the space of compounds by abstracting entity tokens (e.g., "tokyo"→entity). Fig. 3 shows an example tree with two compounds.

For a sampled set $\hat{\mathcal{D}}^{Syn}_{train}$, we use $p(a)$ to denote the frequency distribution of atoms in $\hat{\mathcal{D}}^{Syn}_{train}$, and $p(c)$ to denote the weighted frequency distribution of compounds in $\hat{\mathcal{D}}^{Syn}_{train}$. The compounds are weighted following Keysers et al. (2020), to avoid double-counting of compounds that mostly co-occur with their super-compounds.

**Constructing $\hat{\mathcal{D}}^{Syn}_{train}$:** Let $H(C) = -\sum_c p(c) \log p(c)$ be the entropy over compounds (and similarly for the atom entropy, $H(A)$). Our goal is to find a synthetic training set, $\hat{\mathcal{D}}^{Syn}_{train}$ that maximizes $H(C) + H(A)$.

Finding the subset that maximizes the above objective is computationally hard, hence we use a greedy heuristic, termed CMAXENT, to approximate it. Despite its simplicity, we show in §4 that this approach improves entropy compared to a random sample. Specifically, we start with an empty training set, and in each iteration go over all examples in $\mathcal{D}^{Syn}_{train}$ (with abstract entities), choose the example that maximizes our objective, and add one of the corresponding non-abstract examples to $\hat{\mathcal{D}}^{Syn}_{train}$. We determine the number of iterations according to the desired target size of $\hat{\mathcal{D}}^{Syn}_{train}$. Fig. 4 illustrates a single iteration in this procedure.

**Hybrid approach** Our last approach combines our two sampling procedures, UAT and CMAXENT. Specifically, in each iteration we sample an abstract template uniformly, and then use the greedy procedure for maximizing entropy over queries that correspond to the sampled abstract template. This results in a process that still improves the maximum entropy objective value but is skewed towards a uniform distribution over abstract templates.

## 4 Experiments

We empirically evaluate the contribution of synthetic data to compositional generalization and the extent to which structurally-diverse sampling can improve data efficiency.

### 4.1 Experimental Setting

**Data** Our work assumes access to a small annotated training set and a large pool of synthetic data, generated from a wide-coverage synchronous grammar. We build on work from Xu et al. (2020a), who created the Schema2QA dataset, which contains natural language utterance-query pairs in the ThingTalk language for 6 domains of the Schmea.org onthology: *restaurants, people, hotels, books, movies, and music*. Moreover, Xu et al. (2020b) presented AutoQA as part of the Genie toolkit[2] for generating synthetic examples for Schema.org from a synchronous grammar (Campagna et al., 2019). To obtain enough data we use the manually-labeled data from all 6 domains as our *annotated data*, and term it NL-SCHEMAORG (see §4.3). We generate 7M synthetic examples using AutoQA, and term it SYN-SCHEMAORG.

We construct an i.i.d split and a compositional split to NL-SCHEMAORG, resulting in one common training set $\mathcal{D}_{train}$, containing only 1,619 examples, a compositional development and test sets, and an i.i.d development and test sets (see Table 4 for statistics). The training set, compositional development set, and compositional test set are all disjoint w.r.t their abstract template (see §2). The i.i.d development and test sets are sampled from the same distribution as $\mathcal{D}_{train}$. We create a compositional split of $\hat{\mathcal{D}}^{Syn}_{train}$, resulting in training/development/test sets that are disjoint in terms of their abstract template according to the templates in the compositional split of NL-SCHEMAORG. We describe the exact procedure for splitting the data in Appendix A.

**Evaluation metric** We evaluate models using exact match accuracy, that is, whether the predicted query is identical to the gold query. We denote accuracy on the compositional and i.i.d splits as $\text{EM}_{comp}$ and $\text{EM}_{iid}$ respectively. We report the average and standard deviation over 15 models, which are obtained by training on 3 different random samples $\hat{\mathcal{D}}^{Syn}_{train}$, each with 5 different random seeds.

| Dataset | Split | # examples (train / dev / test) | # new abstract templates (train / dev / test) |
|---|---|---|---|
| ANNOTATED | iid | 1619 / 180 / 199 | 33 / 1 / 1 |
| | Comp. | 1619 / 188 / 304 | 33 / 19 / 20 |
| SYNTHETIC | iid | 5.8M / - / - | 251 / - / - |
| | Comp. | 5.8M / 6K / 6K | 251 / 11 / 10 |

Table 4: Dataset statistics for the i.i.d split and compositional (comp.) split. *# new abstract templates* indicates the number of abstract templates unseen during training time for the development and test sets, and the total number of abstract templates for the training set.

In all experiments, we use $\text{EM}_{iid}$ on the development set to determine early-stopping and for tuning batch size, learning rate and number of warmup steps (see hyper-parameter values in Appendix C).

**Evaluated models** Our baseline parser is fine-tuned on the training set of NL-SCHEMAORG (without pre-training), and is termed BASELINE. We denote the rest of our experiments by the sampling method used to obtain $\hat{\mathcal{D}}^{Syn}_{train}$, where UNIFORM denotes uniform sampling, UAT denotes abstract template sampling (§3.2), CMAXENT denotes compound maximum entropy sampling (§3.3). We denote the hybrid approach, combining the latter methods, as CMAXENT+UAT.

Importantly, we evaluate the effectiveness of our methods across different sizes of $\hat{\mathcal{D}}^{Syn}_{train}$. Overall, we are interested in the interactions between compositional generalization, i.i.d generalization, sampling method, and the size of the synthetic training set. We are especially interested in the effectiveness of our suggested sampling methods using smaller samples, hence limit the sample size to 120K. We denote the size of $\hat{\mathcal{D}}^{Syn}_{train}$ by concatenating it to the model name, e.g., UAT+5K corresponds to sampling with 5K synthetic examples.

As another baseline, we use GECA (Andreas, 2020) as an alternative source for synthetic data. We use the publicly available code,[3] which takes the training set of NL-SCHEMAORG and augments it with 1,342 new examples. We use these examples as $\hat{\mathcal{D}}^{Syn}_{train}$ in our setup.

### 4.2 Results

Table 5 shows test results for both i.i.d and compositional generalization across all methods and synthetic training set sizes

**Uniform sampling** Large amounts of synthetic data improve $\text{EM}_{iid}$ on natural language data com-

[2] https://github.com/stanford-oval/genie-toolkit

[3] https://github.com/jacobandreas/geca

| Split | Method | Sample Size | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | - | 2K | 5K | 10K | 60K | 120K | 500K | 1M |
| *comp.* | BASELINE | 20.1 ± 2.6 | | | | | | | |
| | GECA | 19.7 ± 1.0 | | | | | | | |
| | UNIFORM | | 14.8 ± 2.7 | 22.2 ± 2.4 | 27.6 ± 3 | 28.6 ± 4.1 | 31.8 ± 3.7 | 35.8 ± 1.3 | 37.7 ± 2.9 |
| | UAT | | **27.8** ±3.1 | **41.4** ± 3.3 | **39.0**±2.4 | **43.1** ± 6.5 | **43.0** ± 4.7 | | |
| | CMAXENT | | 17.0 ± 2.8 | 27.0 ± 3.9 | 31.8 ± 1.8 | 34.9 ± 2.7 | 40.2 ± 1.6 | | |
| | CMAXENT+UAT | | 21.1 ± 4.8 | 26.8 ± 5.1 | 34.9 ± 4.9 | 39.1 ± 5.2 | 40.4 ± 3.1 | | |
| *i.i.d* | BASELINE | 81.2 ± 3.2 | | | | | | | |
| | GECA | 79.0 ± 2.1 | | | | | | | |
| | UNIFORM | | 71.5 ± 3.7 | 78.9 ± 4.4 | 83.0 ± 2.7 | 84.4 ± 2.5 | 85.0 ± 1.9 | **85.9** ± 1.2 | 85.0 ± 1.3 |
| | UAT | | **80.7** ±2.7 | **83.4** ± 3.1 | 83.5 ± 2.0 | **85.7** ± 1.9 | 84.4 ± 1.6 | | |
| | CMAXENT | | 79.6 ± 3.2 | 81.1 ± 4.1 | **83.8** ± 1.2 | 84.2 ± 1.3 | 84.6 ± 2.2 | | |
| | CMAXENT+UAT | | 74.9 ± 6.8 | 78.7 ± 4.4 | 80.9 ± 4.5 | 85.1 ± 1.1 | **85.9** ± 1.8 | | |

Table 5: Test compositional and i.i.d EM for all sampling methods (mean and standard deviation). Our structurally-diverse sampling methods allow us to use 200x less training data while improving $\text{EM}_{comp}$ significantly, and retaining comparable $\text{EM}_{iid}$.

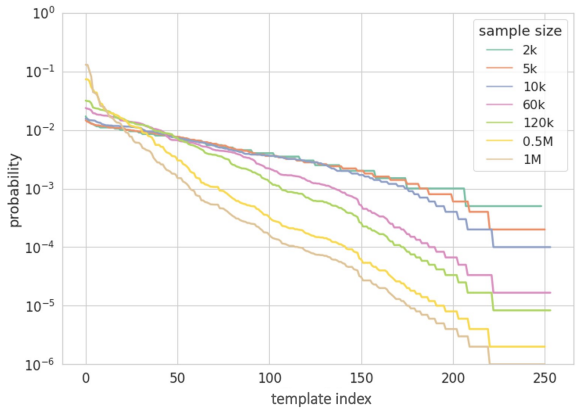


Figure 5: Distribution over abstract templates by sample size. The y-axis is in log-scale. The x-axis enumerates abstract templates sorted by frequency, i.e each point is a template. We observe that when the sample size is smaller, the distribution over abstract templates is more uniform.

pared to BASELINE, which agrees with the findings of Tsarkov et al. (2020). Specifically, with 10K examples $\text{EM}_{iid}$ improves 81.2→83.0, and with 1M examples this improvement reaches 85.0 $\text{EM}_{iid}$. As for compositional generalization, we observe improvements starting from 5K examples, with dramatic gains when training on 1M syntehtic examples – 20.1→37.7 $\text{EM}_{comp}$. To our knowledge, this is the first result showing that randomly sampling synthetic data from a wide-coverage grammar improves compositional generalization on natural language data.

When the size of $\hat{\mathcal{D}}^{Syn}_{train}$ is small, we observe a drop in $\text{EM}_{iid}$, and also in $\text{EM}_{comp}$ when training with 2K examples. This shows that when $\hat{\mathcal{D}}^{Syn}_{train}$ is small, its distribution can potentially adversely affect training.

**Abstract template sampling** UAT sampling dramatically improves compositional generalization even with very little synthetic data. With 2K examples $\text{EM}_{comp}$ improves from 20.1→27.8, and with 5K examples $\text{EM}_{comp}$ is already at 41.4. This is a dramatic improvement compared to uniform sampling – 3.7 $\text{EM}_{comp}$ points higher with *200x less data*.

When further increasing the size of the synthetic data, improvement roughly plateaus, reaching 43.0 $\text{EM}_{comp}$ for 120K examples. A possible explanation for this effect is that as the size of $\hat{\mathcal{D}}^{Syn}_{train}$ grows, the distribution over templates becomes more skewed, as shown in Fig. 5. Changing the composition of $\mathcal{D}^{Syn}_{train}$ to contain more abstract templates by modifying the generation procedure in the AutoQA toolkit, and examining whether this leads to even greater gains in compositional generalization is an important question for future work.

To test if a smooth transition from $\alpha = 1$ to $\alpha = 0$ indeed leads to a smooth transition in compositional generalization, we train models with multiple values of $\alpha$. Fig. 6 confirms that tuning $\alpha$ from 1 to 0 yields a gradual improvement in $\text{EM}_{comp}$.

Last, $\text{EM}_{iid}$ is comparable to UNIFORM, and even higher for smaller samples.

**Compound maximum entropy** CMAXENT improves compositional generalization with greater data efficiency compared to UNIFORM, as it improves $\text{EM}_{comp}$ at all sample sizes. With 120K examples, CMAXENT reaches 40.2 $\text{EM}_{comp}$, and surpasses UNIFORM+1M, at 37.7.

Still, UAT outperforms CMAXENT in all cases. There are several possible explanations for this phenomenon. First, it might be the case that the distribution over abstract templates is the important

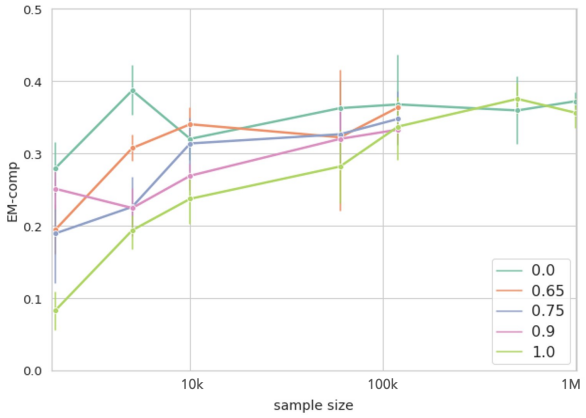


Figure 6: Accuracy on the compositional development set by size and $\alpha$ value. $\alpha = 1$ is equivalent to a uniform sampling over examples, and as $\alpha$ decreases the distribution over abstract templates becomes closer to uniform. We report the average over 5 random seeds, and bars denote 95% confidence intervals. x-axis is in log-scale.

factor in determining compositional generalization. In Appendix D we show that indeed the distribution over abstract templates of CMAXENT is more skewed compared to UAT. Second, our heuristic for optimizing entropy might be sub-optimal. While we do observe an entropy increase from 6.1→7.1 in compound entropy, and from 3.9→4.4 in atom entropy, it is possible that a better optimization procedure or a better definition of the notion of *compound* will yield further gains in compositional generalization. We leave this investigation for future work.

**Hybrid sampling** Combining CMAXENT and UAT leads to improvements in $\text{EM}_{comp}$ over CMAXENT for all sample sizes (except 5K), but the overall performance does not surpass UAT.

**GECA** Results are slightly lower than BASELINE, 79.0 $\text{EM}_{iid}$ and 19.7 $\text{EM}_{comp}$. Sampling 1,342 examples from GECA is better than UNIFORM+2K, but worse than UAT+2K. Thus, an advantage in the synchronous grammar is that we can easily sample a large number of synthetic examples that cover a wider range of structures.

NL-SCHEMAORG comprises data from 6 different domains. In Table 10 in Appendix F, we show development EM per domain. While the number of examples in each domain is small (a few dozen examples per domain), we still observe similar trends across all domains.

To summarize, our results suggest that abstract template diversity is an important factor in compositional generalization, and generating synthetic data with many abstract templates can dramatically improve compositional generalization.

### 4.3 Analysis

We perform a manual analysis of models' predictions on NL-SCHEMAORG compositional development set. We inspect 40 predictions of 8 models, and identify three major error types. The first type is *structural* errors, which include syntax errors, misplacing parenthesis, and incorrect typing of properties. The second group is *linking* errors, including, e.g., hallucination of entities, return of unsorted results, and missing an aggregation step. Third, we identify errors that are *benign*, where the predicted queries are valid and equivalent to the target query. An example is using a wrong operator that does not change the query. We also measure robustness to DB entity replacement in a query by grouping together any subset of development examples that only differ in a DB entity, and counting for how many groups all the examples are predicted correctly.

Table 6 shows the results of the analysis. Our findings suggest two main benefits of larger $\hat{\mathcal{D}}^{Syn}_{train}$: (a) more frequently, the errors are benign, and (b) generalization is more robust to DB entity replacement. In addition, we find that using UAT reduces structural errors, but increases linking errors, e.g, missing necessary `sort` or `filter` steps. Last, linking errors are the most common error type across all models.

Inspecting the predictions of UNIFORM+1M and UAT+5K on the development set, we find that the abstract templates of correct predictions constitute roughly 40% of the templates in the set, and are almost identical between the two models. We notice that "hard" templates are not necessarily longer or more nested.

## 5 Related Work

**Data augmentation** Previous work studied different data augmentation techniques to improve i.i.d generalization in semantic parsing including synchronous grammars (Jia and Liang, 2016; Yu et al., 2020; Xu et al., 2020b), target side grammars with neural generation models (Tran and Tan, 2020; Wang et al., 2021), and pre-training with auxiliary tasks (Yin et al., 2020; Deng et al., 2021). In the context of compositional generalization, data aug-

| Method | (a) Benign Errors | Linking Errors | Structural Errors | (b) Consistent Queries |
|---|---|---|---|---|
| UNIFORM | | | | |
| +5k | 7 | 63 | 30 | 15 |
| +60k | 8 | 84 | 8 | 27 |
| +120k | 10 | 76 | 14 | 31 |
| +1M | 11 | 63 | 26 | 37 |
| CMAXENT | | | | |
| +5k | 8 | 62 | 31 | 24 |
| +60k | 9 | 52 | 39 | 31 |
| UAT | | | | |
| +5k | 8 | 92 | 0 | 33 |
| +60k | 0 | 91 | 9 | 32 |

Table 6: Error analysis. (a) a categorization of 40 predictions on the compositional development set, selected at random. Errors are partitioned by similar patterns to: **Benign errors**: the prediction is different but semantically equivalent to the target query, **Linking error**: a mismatch between information in input utterance and predicted query, and **Structural error**: wrong use of parenthesis and invalid queries. (b) the percentage of queries that are predicted correctly for any DB entity that occurs in the development set.

mentation was achieved by re-combining training examples in new ways (Andreas, 2020; Akyürek et al., 2021), or by back-translation (Guo et al., 2020c). Conversely, we generate data from an independent wide-coverage grammar and investigate data-efficient sampling through structured diversity.

Outside of semantic parsing, it has been shown in a grounded learning setup (Hill et al., 2020) that increasing lexical diversity can improve out-of-distribution generalization.

**Compositional Generalization** In contrast to our work that focuses on sampling synthetic data, many other approaches have been suggested to improve compositional generalization in semantic parsing. These include new or modified model architectures (Li et al., 2019; Gordon et al., 2020; Guo et al., 2020b; Oren et al., 2020; Zheng and Lapata, 2020; Herzig and Berant, 2021; Shaw et al., 2020), pre-trained language models (Furrer et al., 2020), intermediate representations (Herzig et al., 2021), and meta learning (Lake, 2019; Conklin et al., 2021).

**Data Selection** Our work is related to algorithmic approaches for reducing biases in datasets such as adversarial filtering (Le Bras et al., 2020; Sakaguchi et al., 2020) and representation debiasing (Li and Vasconcelos, 2019; Li et al., 2018). Our approach utilizes the structural nature of executable queries, and focuses on biases related to structural diversity.

## 6 Conclusion

In this work, we for the first time explore whether generating large amounts of synthetic data from a synchronous grammar improves compositional generalization, and propose sampling methods that allow for more efficient training by generating structurally-diverse training sets. We find that synthetic data dramatically improves generalization, and moderately improves i.i.d generalization, and that by uniformly sampling abstract templates, we can improve data efficiency by a factor of 200x.

In the past year, a myriad of approaches have been proposed for encouraging compositional generalization through modeling innovations, clever training procedures, and data augmentation techniques. Our works adds to the body of work that shows that data augmentation is an effective strategy even with small amounts of augmented data, when examples are carefully constructed. Moreover, data augmentation techniques can be easily combined with new models and training procedures, potentially leading to further gains in compositional generalization.

In addition, we believe our findings can be generalized to other NLP tasks to improve data efficiency w.r.t both i.i.d and compositional generalization. This requires defining a structure over the training examples that can be used similar to the structures used in this work.

## Acknowledgements

We thank Elad Segal, Ben Bogin, Matan Hasson and Ori Yoran for their useful suggestions. This research was supported in part by The Yandex Initiative for Machine Learning, and The European Research Council (ERC) under the European Union Horizons 2020 research and innovation programme (grant ERC DELPHI 802800). The second author was supported by a Google PhD fellowship.

## References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations (ICLR)*.

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.

Giovanni Campagna, Silei Xu, Mehrad Moradshahi, Richard Socher, and Monica S Lam. 2019. Genie: A generator of natural language semantic parsers for virtual assistant commands. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 394–410.

Jianpeng Cheng, Siva Reddy, and Mirella Lapata. 2018. Building a neural semantic parser from a domain ontology. *arXiv preprint arXiv:1812.10037*.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3322–3335, Online. Association for Computational Linguistics.

Xiang Deng, Ahmed H. Awadallah, Chris Meek, Alex Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-sql. In *NAACL 2021*.

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. 2020. Permutation equivariant models for compositional generalization in language. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yu Gu, Sue Kase, Michelle T. Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *World Wide Web (WWW)*.

Demi Guo, Yoon Kim, and Alexander Rush. 2020a. Sequence-level mixed sample data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5547–5552, Online. Association for Computational Linguistics.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020b. Hierarchical poset decoding for compositional generalization in language. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2020c. Revisiting iterative back-translation from the perspective of compositional generalization. *arXiv preprint arXiv:2012.04276*.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Association for Computational Linguistics (ACL)*.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *arXiv preprint arXiv:2104.07478*.

Felix Hill, Andrew K. Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. 2020. Environmental drivers of systematicity and generalization in a situated agent. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 9791–9801. Curran Associates, Inc.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *International Conference on Machine Learning*, pages 1078–1088. PMLR.

Dongjun Lee, Jaesik Yoon, Jongyun Song, Sanggil Lee, and Sungroh Yoon. 2019. One-shot learning for text-to-sql generation. *arXiv preprint arXiv:1905.11499*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yi Li and Nuno Vasconcelos. 2019. Repair: Removing representation bias by dataset resampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9572–9581.

Yingwei Li, Yi Li, and Nuno Vasconcelos. 2018. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528.

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 4293–4302, 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. *arXiv preprint arXiv:2006.10627*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*, 63.

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

Ke Tran and Ming Tan. 2020. Generating synthetic data for task-oriented semantic parsing with hierarchical representations. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 17–21, Online. Association for Computational Linguistics.

Dmitry Tsarkov, Tibor Tihon, Nathan Scales, Nikola Momchev, Danila Sinopalnikov, and Nathanael Schärli. 2020. *-cfq: Analyzing the scalability of machine learning on a compositional task. *arXiv preprint arXiv:2012.08266*.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to synthesize data for semantic parsing. *arXiv preprint arXiv:2104.05827*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. 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 1332–1342, Beijing, China. Association for Computational Linguistics.

Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, Ugur Çetintemel, and Carsten Binnig. 2020. Dbpal: A fully pluggable NL2SQL training pipeline. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 2347–2361. ACM.

Silei Xu, Giovanni Campagna, Jian Li, and Monica S. Lam. 2020a. Schema2qa: High-quality and low-cost q&a agents for the structured web. In *CIKM '20: The 29th ACM International Conference on*

*Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 1685–1694. ACM.

Silei Xu, Sina Semnani, Giovanni Campagna, and Monica Lam. 2020b. AutoQA: From databases to QA semantic parsers with only synthetic training data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 422–434, Online. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

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

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

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

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6869–6882, Online. Association for Computational Linguistics.

| Dataset | Domain | # examples | # abstract templates |
|---|---|---|---|
| ANNOTATED | books | 356 | 21 |
| | hotels | 438 | 24 |
| | movies | 379 | 23 |
| | music | 310 | 14 |
| | people | 494 | 27 |
| | restaurants | 513 | 44 |
| SYNTHETIC | books | 1,022,222 | 107 |
| | hotels | 477,759 | 151 |
| | movies | 1,101,019 | 112 |
| | music | 1,277,895 | 108 |
| | people | 1,339,201 | 63 |
| | restaurants | 637,179 | 121 |

Table 7: Domain distribution in the synthetic and annotated datasets.

| Size | UNIFORM | UAT | CMAXENT | UAT +CMAXENT |
|---|---|---|---|---|
| 2k | 70 | 256 | 136 | 192 |
| 5k | 77 | 258 | 156 | 247 |
| 10k | 95 | 261 | 214 | 254 |
| 60k | 152 | 261 | 246 | 258 |
| 120k | 169 | 261 | 252 | 258 |
| 0.5M | 193 | | | |
| 1M | 223 | | | |

Table 8: Number of abstract templates seen during pre-training and fine-tuning by size and sampling method.

## A Data Split Procedure

To construct an i.i.d split and a compositional split we perform the following procedure. First, we construct a compositional split to NL-SCHEMAORG, resulting in 3 sets that are disjoint in terms of their abstract template. We use 2 of them as the compositional development set and the compositional test set. The third set is split i.i.d to trainining set, i.i.d development set, and i.i.d test set. Next, we select the training, compositional development and compositional test sets from SYN-SCHEMAORG. We assign to each set the examples that their abstract template appear in the corresponding set of NL-SCHEMAORG. Examples with a new abstract template (a template that do not occur in NL-SCHEMAORG), are also included in the training set. We downsample the compositional development and test sets of SYN-SCHEMAORG to 6K examples.

## B Domain Statistics

Table 7 shows the domain distribution in the synthetic and annotated datasets.

## C Training

We implement and train our models using AllenNLP with PyTorch as backend, and initialize them using BART base. We conduct experiments on a machine with 8 NVIDIA GeForce GTX 2080 GPUs and 40 Intel(R) Xeon(R) Silver 4114 CPUs. The OS is Ubuntu 18.04.3 LTS.

**Hyper-parameters** We use the Adam optimizer (Loshchilov and Hutter, 2019) with learning rate selected from $\{0.00001, 0.00002, 0.00003\}$. Batch size is selected from $\{1, 8\}$ for sample size $\leq$ 5000, and $\{24, 48, 64\}$ for larger samples. We use a learning rate schedueler with polynomial decay and select warm up steps from $\{1000, 1500, 2000\}$ for sample sizes $\leq 1000$, and $\{2500, 3000, 3500, 4000\}$ for larger samples. We use patience of 5 epochs in pre-training, and 10 epochs in fine-tuning. We use EM on the i.i.d development set as a metric for early stopping and selecting the best hyper-parameters. Hyper-parameters are fine-tuned for each sampling method and sample size separately on a single sample of SYN-SCHEMAORG. The patience is selected from $\{5, 8, 10\}$ when training on samples different than the one used for fine-tuning. Hyper-parameters for the models fine-tuned on NL-SCHEMAORG are fine-tuned once on a UNIFORM sample.

## D Sample Diversity

We measure structural-diversity in terms of abstract template distribution, atoms entropy and compounds entropy. Table 8 compares the total number of abstract templates seen during both pre-training and fine-tuning between sampling methods and sample sizes. Figure 7 compares the normalized frequency of abstract templates between sampling methods for two sample sizes. Figure 8 compares the atoms and compounds entropy between sampling methods and sample sizes. The above statistics are on a single sample from each method and size.

## E Generalization of Synthetic vs. Annotated Data

Figure. 9 shows for each sampling method and sampling size, the $\text{EM}_{comp}$ of the pre-trained model on SYN-SCHEMAORG development set, and the $\text{EM}_{comp}$ of the fine-tuned model on NL-SCHEMAORG development set. The relation between the performances is correlated for sample

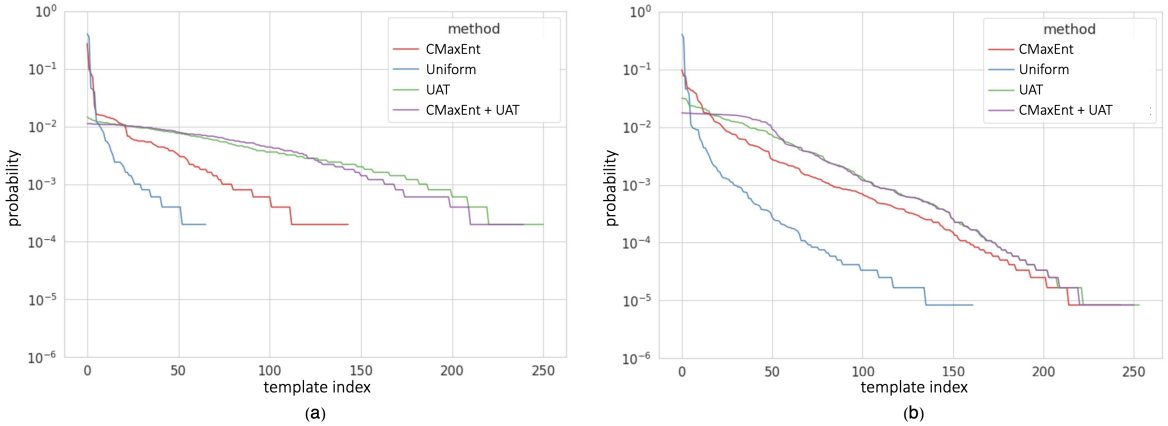


Figure 7: Distribution over abstract templates by sampling method, for (a) a sample of size 5K, and (b) a sample of size 120K. The y-axis is in log-scale. The x-axis enumerates abstract templates sorted by probability, i.e each point is a template.

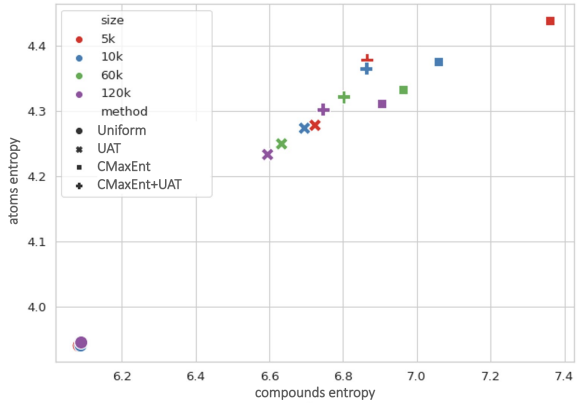


Figure 8: Atoms and compounds entropy by sample size and sampling method, for one sample.

sizes smaller than 10K. We report the average over 5 random seeds and a single sample.

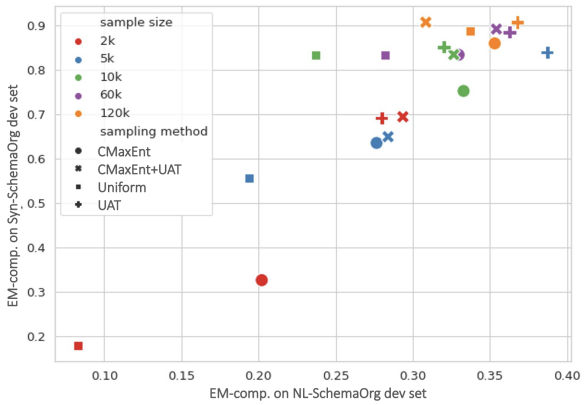


Figure 9: $EM_{comp}$ of the pre-trained model on SYN-SCHEMAORG development set, and the $EM_{comp}$ of the fine-tuned model on NL-SCHEMAORG development set. We report the average over 5 random seeds on a single sample.

## F Development Results

Table 9 contains the NL-SCHEMAORG development set $EM_{comp}$ and $EM_{iid}$ for all sampling methods and sampling sizes. Table 10 shows the development set results by domain.

| Split | Method | - | Sample Size | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | 2k | 5k | 10k | 60k | 120k | 500k | 1M |
| *comp.* | BASELINE | 13.1 | | | | | | | |
| | GECA | 8.7 | | | | | | | |
| | UNIFORM | | 6.6 | 17.0 | 22.5 | 30.7 | 32.7 | 36.7 | 36.5 |
| | UAT | | 28.3 | 36.1 | 34.2 | 35.2 | 37.0 | | |
| | CMAXENT | | 19.5 | 22.9 | 32.8 | 31.7 | 32.5 | | |
| | CMAXENT+UAT | | 19.6 | 36.2 | 31.8 | 31.7 | 32.3 | | |
| *i.i.d* | BASELINE | 81.7 | | | | | | | |
| | GECA | 82.1 | | | | | | | |
| | UNIFORM | | 76.6 | 82.1 | 83.9 | 86.0 | 87.0 | 88.8 | 88.8 |
| | UAT | | 81.6 | 83.3 | 85.1 | 86.5 | 87.8 | | |
| | CMAXENT | | 80.7 | 81.2 | 85.0 | 87.3 | 87.8 | | |
| | CMAXENT+UAT | | 78.4 | 81.8 | 79.0 | 87.2 | 87.2 | | |

Table 9: Development EM for all sampling methods. We report average over 15 models, which are obtained by training on 3 different samples, each with 5 different random seeds.

| Split | Domain | #Examples | Method | Sample Size 2k | 5k | 10k | 60k | 120k | 500k | 1M |
|---|---|---|---|---|---|---|---|---|---|---|
| *comp.* | Books | 21 | UNIFORM | 16.2 | 21.9 | 21.9 | 32.4 | 39.7 | 40.9 | 34.1 |
| | | | UAT | 31.4 | 47.6 | 35.2 | 48.6 | 40.5 | | |
| | | | CMAXENT | 20.9 | 30.5 | 43.8 | 29.5 | 39.1 | | |
| | | | CMAXENT+UAT | 44.8 | 29.5 | 39.1 | 42.9 | 35.2 | | |
| | Hotels | 28 | UNIFORM | 2.9 | 15.7 | 24.3 | 26.4 | 35.7 | 50.0 | 50.6 |
| | | | UAT | 27.9 | 46.4 | 39.3 | 47.1 | 41.1 | | |
| | | | CMAXENT | 21.4 | 26.4 | 41.4 | 40.7 | 42.1 | | |
| | | | CMAXENT+UAT | 30.7 | 19.3 | 30.0 | 42.1 | 44.3 | | |
| | Movies | 33 | UNIFORM | 15.2 | 27.9 | 27.9 | 27.3 | 40.4 | 37.6 | 33.8 |
| | | | UAT | 27.9 | 53.3 | 30.9 | 41.2 | 40.4 | | |
| | | | CMAXENT | 22.4 | 32.7 | 34.5 | 31.5 | 36.4 | | |
| | | | CMAXENT+UAT | 31.5 | 32.7 | 35.1 | 38.2 | 32.7 | | |
| | Music | 31 | UNIFORM | 0.0 | 0.7 | 7.1 | 13.6 | 13.4 | 11.6 | 11.8 |
| | | | UAT | 4.5 | 17.4 | 5.2 | 9.7 | 15.0 | | |
| | | | CMAXENT | 6.5 | 11.0 | 9.0 | 10.3 | 11.0 | | |
| | | | CMAXENT+UAT | 12.9 | 16.1 | 17.4 | 9.7 | 14.2 | | |
| | People | 23 | UNIFORM | 6.1 | 9.6 | 6.1 | 9.6 | 15.2 | 10.4 | 9.4 |
| | | | UAT | 13.9 | 15.7 | 15.7 | 12.2 | 13.0 | | |
| | | | CMAXENT | 8.7 | 7.0 | 9.6 | 8.7 | 10.4 | | |
| | | | CMAXENT+UAT | 8.7 | 8.7 | 7.8 | 8.7 | 13.0 | | |
| | Restaurants | 52 | UNIFORM | 9.6 | 30.4 | 39.2 | 45.0 | 49.7 | 57.3 | 49.4 |
| | | | UAT | 46.9 | 44.6 | 50.8 | 48.9 | 54.2 | | |
| | | | CMAXENT | 31.1 | 43.1 | 48.9 | 55.4 | 55.0 | | |
| | | | CMAXENT+UAT | 40.0 | 46.2 | 50.0 | 54.2 | 38.5 | | |
| *i.i.d* | Books | 25 | UNIFORM | 79.2 | 83.2 | 84.0 | 83.2 | 87.3 | 88.0 | 88.7 |
| | | | UAT | 82.4 | 84.8 | 86.4 | 86.4 | 88.0 | | |
| | | | CMAXENT | 83.2 | 85.6 | 84.8 | 86.4 | 87.2 | | |
| | | | CMAXENT+UAT | 86.4 | 76.0 | 81.6 | 87.2 | 88.0 | | |
| | Hotels | 33 | UNIFORM | 73.3 | 84.2 | 87.3 | 83.0 | 86.4 | 87.9 | 85.9 |
| | | | UAT | 76.4 | 80.0 | 83.0 | 84.9 | 83.3 | | |
| | | | CMAXENT | 75.8 | 82.4 | 84.2 | 89.7 | 90.3 | | |
| | | | CMAXENT+UAT | 83.0 | 78.2 | 72.7 | 87.9 | 87.9 | | |
| | Movies | 32 | UNIFORM | 77.5 | 85.0 | 83.1 | 83.1 | 86.5 | 90.6 | 86.5 |
| | | | UAT | 83.8 | 86.2 | 91.2 | 85.0 | 88.0 | | |
| | | | CMAXENT | 85.6 | 85.0 | 89.4 | 88.8 | 87.5 | | |
| | | | CMAXENT+UAT | 83.1 | 74.4 | 88.1 | 89.4 | 90.0 | | |
| | Music | 12 | UNIFORM | 71.7 | 78.3 | 85.0 | 81.7 | 80.6 | 85.0 | 91.7 |
| | | | UAT | 73.3 | 85.0 | 88.3 | 81.7 | 87.5 | | |
| | | | CMAXENT | 81.7 | 85.0 | 91.7 | 86.7 | 88.3 | | |
| | | | CMAXENT+UAT | 85.0 | 78.3 | 90.0 | 88.3 | 86.7 | | |
| | People | 45 | UNIFORM | 84.4 | 89.8 | 88.4 | 87.6 | 90.0 | 88.4 | 90.7 |
| | | | UAT | 86.2 | 88.0 | 85.8 | 88.0 | 90.0 | | |
| | | | CMAXENT | 84.9 | 90.7 | 85.8 | 90.7 | 89.8 | | |
| | | | CMAXENT+UAT | 91.1 | 87.6 | 82.7 | 90.2 | 89.3 | | |
| | Restaurants | 33 | UNIFORM | 80.0 | 87.9 | 79.4 | 85.5 | 87.9 | 92.1 | 88.9 |
| | | | UAT | 76.4 | 83.6 | 84.9 | 81.2 | 88.4 | | |
| | | | CMAXENT | 80.0 | 80.6 | 84.2 | 85.5 | 89.1 | | |
| | | | CMAXENT+UAT | 86.1 | 81.8 | 81.8 | 85.5 | 82.4 | | |

Table 10: Development EM for all sampling methods, by domain. We report average over 15 models, which are obtained by training on 3 different samples, each with 5 different random seeds.