On The Ingredients of an Effective Zero-shot Semantic Parser

Pengcheng Yin♠* **John Wieting**♣ **Avirup Sil**♦ **Graham Neubig**♠
♠Carnegie Mellon University ♣Google Research ♦IBM Research AI
{pcyin,gneubig}@cs.cmu.edu jwieting@google.com avi@us.ibm.com

Abstract

Semantic parsers map natural language utterances into meaning representations (*e.g.* programs). Such models are typically bottlenecked by the paucity of training data due to the laborious annotation efforts. Recent studies have performed zero-shot learning by synthesizing training examples of canonical utterances and programs from a grammar, and further paraphrasing these utterances to improve linguistic diversity. However, such synthetic examples cannot fully capture patterns in real data. In this paper we analyze zero-shot parsers through the lenses of the *language* and *logical* gaps (Herzig and Berant, 2019), which quantify the discrepancy of language and programmatic patterns between the synthetic canonical examples and real-world user-issued ones. We propose bridging these gaps using improved grammars, stronger paraphrasers, and efficient learning methods using canonical examples that most likely reflect real user intents. Our model achieves strong results on the SCHOLAR and GEO benchmarks with *zero* labeled data.[1]

1 Introduction

Semantic parsers translate natural language (NL) utterances into formal meaning representations. In particular, task-oriented semantic parsers map user-issued utterances (*e.g. Find papers in ACL*) into machine-executable programs (*e.g.* a database query), play a key role in providing natural language interfaces to applications like conversational virtual assistants (Gupta et al., 2018; Andreas et al., 2020), robot instruction following (Artzi and Zettlemoyer, 2013; Fried et al., 2018), as well as querying databases (Li and Jagadish, 2014; Yu et al., 2018) or generating Python code (Yin and Neubig, 2017).

Learning semantic parsers typically requires parallel data of utterances annotated with programs, which requires significant expertise and cost (Berant et al., 2013). Thus, the field has explored alternative approaches using supervisions cheaper to acquire, such as the execution results (Clarke et al., 2010) or unlabeled utterances (Poon, 2013). In particular, the seminal OVERNIGHT approach (Wang et al., 2015) synthesizes parallel data by using a synchronous grammar to align programs and their canonical NL expressions (*e.g.* `Filter(paper,venue=`$\boxed{?}$`)` ↔ *papers in* $\boxed{?}$ and `acl`↔*ACL*), then generating examples of compositional utterances (*e.g. Papers in ACL*) with programs (*e.g.* `Filter(paper,venue=acl)`). The synthesized utterances are paraphrased by annotators, a much easier task than writing programs.

Recently, Xu et al. (2020b) build upon OVERNIGHT and develop a *zero-shot* semantic parser replacing the manual paraphrasing process with an automatic paraphrase generator (§2). While promising, there are still several open challenges. First, such systems are not truly zero-shot — they still require labeled validation data (*e.g.* to select the best checkpoint at training). Next, to ensure the quality and broad-coverage of synthetic canonical examples, existing models rely on heavily curated grammars (*e.g.* with 800 production rules), which are cumbersome to maintain. More importantly, as suggested by Herzig and Berant (2019) who study OVERNIGHT models using manual paraphrases, such systems trained on synthetic samples suffer from fundamental mismatches between the distributions of the automatically generated examples and the *natural* ones issued by real users. Specifically, there are two types of gaps. First, there is a *logical gap* between the synthetic and real programs, as real utterances (*e.g. Paper coauthored by Peter and Jane*) may exhibit logic patterns outside of the domain of those covered by the grammar (*e.g. Paper by Jane*). The second is the *language gap* between the synthetic and real utterances, as paraphrased utterances (*e.g.* u'_1 in Fig. 1) still follow similar linguistic patterns as the canonical ones they are

*Now at Google Brain. Email to pcyin@google.com.
[1] https://pcyin.me/zeroshot_parser/

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1455 - 1474

paraphrased from (*e.g.* $\boldsymbol{u}_1$), while user-issued utterances are more linguistically diverse (*e.g.* $\boldsymbol{u}_2$).

In this paper we analyze zero-shot parsers through the lenses of language and logical gaps, and propose methods to close those gaps (§3). Specifically, we attempt to bridge the language gap using stronger paraphrasers and more expressive grammars tailored to the domain-specific idiomatic language patterns. We replace the large grammars of previous work with a highly compact grammar with only 46 domain-general production rules, plus a small set of domain-specific productions to capture idiomatic language patterns (*e.g.* $\boldsymbol{u}_2$ in Fig. 1, §3.1.1). We demonstrate that models equipped with such a smaller but more expressive grammar catered to the domain could generate utterances with more idiomatic and diverse language styles.

On the other hand, closing the logical gap is non-trivial, since canonical examples are generated by exhaustively enumerating all possible programs from the grammar up to a certain depth, and increasing the threshold to cover more complex real-world examples will lead to exponentially more canonical samples, the usage of which is computationally intractable. To tackle the exponentially exploding sample space, we propose an efficient sampling approach by retaining canonical samples that most likely appear in real data (§3.1.2). Specifically, we approximate the likelihood of canonical examples using the probabilities of their utterances measured by pre-trained language models (LMs). This enables us to improve logical coverage of programs while maintaining a tractable number of highly-probable examples as training data.

By bridging the language and logical gaps, our system achieves strong results on two datasets featuring realistic utterances (SCHOLAR and GEO). Despite the fact that our model uses *zero* annotated data for training and validation, it outperforms other supervised methods like OVERNIGHT and GRANNO (Herzig and Berant, 2019) requiring manual annotation. Analysis shows that current models are far from perfect, suggesting logical gap still remains an issue, while stronger paraphrasers are needed to further close the language gap.

2 Zero-shot Semantic Parsing via Data Synthesis

Problem Definition Semantic parsers translate a user-issued NL utterance $\boldsymbol{u}$ into a machine-executable program $\boldsymbol{z}$ (Fig. 1). We consider a zero-shot learning setting without access to parallel data in the target domain. Instead, the system is trained on a collection of machine-synthesized examples.

Overview Our system is inspired by the existing zero-shot semantic parser AUTOQA (Xu et al., 2020b). Fig. 1 illustrates our framework. Intuitively, we automatically create training examples with canonical utterances from a grammar, which are then paraphrased to increase diversity in language style. Specifically, there are two stages. First, a set of seed canonical examples (Fig. 1b) are generated from a **synchronous grammar**, which defines compositional rules of NL expressions to form utterances (Fig. 1a). Next, in the iterative training stage, a **paraphrase generation** model rewrites the canonical utterances to more natural and linguistically diverse alternatives (Fig. 1c). The paraphrased examples are then used to train a semantic parser. To mitigate noisy paraphrases, a filtering model, which is the parser trained on previous iterations, rejects paraphrases that are potentially incorrect. This step of paraphrasing and training could proceed for multiple iterations, with the parser trained on a dataset with growing diversity of language styles.[2]

Synchronous Grammar Seed canonical examples are generated from a synchronous context free grammar (SCFG). Fig. 1a lists simplified production rules in the grammar. Intuitively, productions specify how utterances are composed from lower-level language constructs and domain lexicons. For instance, given a database entity `alan_turing` with a property `citations`, $\boldsymbol{u}_3$ in Fig. 1 could be generated using r_1. Productions could be applied recursively to derive more compositional utterances (*e.g.* $\boldsymbol{u}_2$ using r_2, r_4 and r_6). Our SCFG is based on Herzig and Berant (2019), consisting of domain-general rules of generic logical operations (*e.g.* `superlative`, r_3) and domain-specific lexicons of entity types and relations. Different from Xu et al. (2020b) which uses a complex grammar with 800 rules, we use a compact grammar with only 46 generic rules plus a handful of idiomatic productions (§3.1.1) to capture domain-specific language patterns (*e.g.* *"most recent"* in $\boldsymbol{u}_2$, *c.f.*, $\boldsymbol{u}_1$). Given the grammar, examples are enumerated exhaustively up to a threshold of number of rule applications, yielding a large set of seed canonical

[2]This process is similar to expert iteration in reinforcement learning (Anthony et al., 2017), where a model is iteratively re-trained on newly discovered action trajectories.

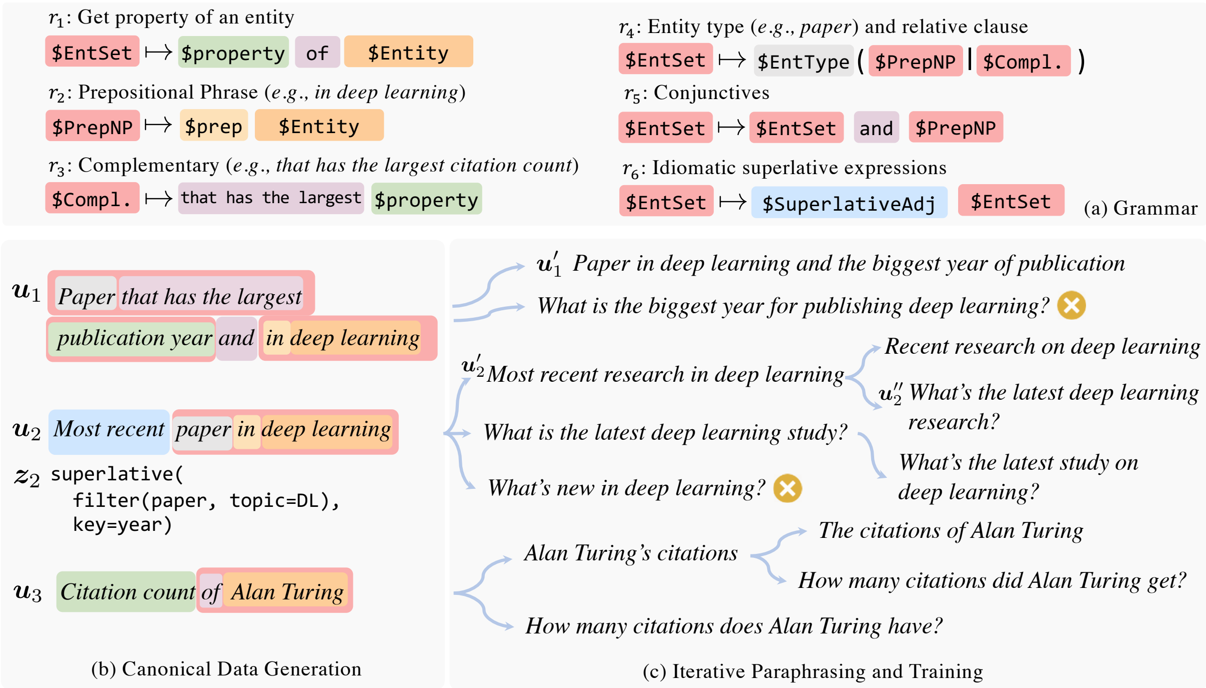


Figure 1: Illustration of the learning process of our zero-shot semantic parser with real model outputs. **(a)** Synchronous grammar with production rules. **(b)** Canonical examples of utterances with programs (only z_2 is shown) are generated from the grammar (colored spans show productions used). Programs are shown in simplified, illustrative form. Refer to Appendix B for real examples. Unnatural utterances like u_1 can be discarded, as in §3.1.2 **(c)** At each iteration, canonical examples are paraphrased to increase diversity in language style, and a semantic parser is trained on the paraphrased examples. Potentially noisy or vague paraphrases are filtered (marked as ⊗) using the parser trained on previous iterations.

examples $\mathbb{D}_{\text{can}}$ (Fig. 1b) for paraphrasing.[3]

Paraphrase Generation and Filtering The paraphrase generation model rewrites a canonical utterance u to more natural and diverse alternatives u'. u' is then paired with u's program to create a new example. We finetune a BART model on the dataset by Krishna et al. (2020), which is a subset of the PARANMT corpus (Wieting and Gimpel, 2018) that contain lexically and syntactically diverse paraphrases. The model therefore learns to produce paraphrases with diverse linguistic patterns, which is essential for closing the language gap when paraphrasing from canonical utterances. To further improve the syntactic diversity of paraphrases from imperative utterances (*e.g.* u_2, Fig. 1), we apply forced decoding such that half of the generated paraphrases start with questions with WH-prefixes (*e.g.* u_3 in Fig. 1). Refer to Appendix A for details. Still, some paraphrases are noisy or potentially vague (⊗ in Fig. 1c). We follow Xu et al. (2020b) and use the parser trained on previous iterations as the filtering model, and reject paraphrases for which the parser cannot predict their programs.

[3] SCFGs could not generate utterances with context-dependent rhetorical patterns such as anaphora. Our model could still handle simple domain-specific context-dependent patterns (*e.g. Paper by A and B*, where *A* and *B* are different authors) by first generating all the canonical samples and then filtering those that violate the constraints.

3 Bridging the Gaps between Canonical and Natural Data

Language and Logical Gaps The synthesis approach in §2 will yield a large set of paraphrased canonical data (denoted as $\mathbb{D}_{\text{par}}$). However, as noted by Herzig and Berant (2019) (hereafter HB19), the synthetic examples cannot capture all the language and programmatic patterns of real-world natural examples from users (denoted as $\mathbb{D}_{\text{nat}}$). There are two mismatches between $\mathbb{D}_{\text{par}}$ and $\mathbb{D}_{\text{nat}}$. First, there is a **logical gap** between real programs in $\mathbb{D}_{\text{nat}}$ and the synthetic ones in $\mathbb{D}_{\text{par}}$, which are exhaustively composed up to a certain compositional depth and therefore cannot capture more complex programs in $\mathbb{D}_{\text{nat}}$. Next, there is a **language gap** between paraphrased canonical utterances and real-world user-issued ones. Real utterances (*e.g.* u_2 in Fig. 1, which is from $\mathbb{D}_{\text{nat}}$ but can be modeled as a canonical sample later in §3.1.1) enjoy more lexical and syntactic diversity, while the auto-paraphrased ones (*e.g.* u'_1) are typically biased towards the clunky language style of their canonical source (*e.g.* u_1). While we could increase diversity via iterative rounds of paraphrasing (*e.g.* $\boldsymbol{u}_2 \mapsto \boldsymbol{u}'_2 \mapsto \boldsymbol{u}''_2$), the paraphraser could still fail on canonical utterances that are not natural English sentences at all, like $\boldsymbol{u}_1$.

3.1 Bridging Language and Logical Gaps

We introduce improvements to the system to close the language (§3.1.1) and logical (§3.1.2) gaps.

3.1.1 Idiomatic Productions

To close language gaps, we augment the grammar with productions capturing domain-specific idiomatic language styles. Such productions compress the clunky canonical expressions (*e.g.* $\boldsymbol{u}_1$ in Fig. 1) to more succinct and natural alternatives (*e.g.* $\boldsymbol{u}_2$), inspired by prior studies on how human experts revise canonical utterances (Wang et al., 2015), as well as by studying samples in real data. Specifically, we focus on two language patterns:

Non-compositional expressions for multi-hop relations Compositional canonical utterances typically feature chained multi-hop relations that are joined together (*e.g. Author that writes paper whose topic is NLP*), which can be compressed using more succinct phrases to denote the relation chain, where the intermediary pivoting entities (*e.g.* `paper`) are omitted (*e.g. Author that works on NLP*). The pattern is referred to as sub-lexical compositionality in Wang et al. (2015) and used by annotators to compress verbose canonical utterances, while we model them using grammar rules. Refer to Appendix B for more details.

Idiomatic Comparatives and Superlatives The general grammar in Fig. 1a uses canonical constructs for comparative (*e.g. smaller than*) and superlative (*e.g. largest*) utterances (*e.g.* $\boldsymbol{u}_1$), which is not ideal for entity types with special units (*e.g.* time, length). We therefore create productions specifying idiomatic comparative and superlative expressions (*e.g. paper published before 2014*, and $\boldsymbol{u}_2$ in Fig. 1). Sometimes, answering a superlative utterance requires reasoning with other pivoting entities. For instance, the relation in *"venue that X publish mostly in"* between authors and venues implicitly involves counting the papers that *X* publishes. For such cases, we create "macro" productions, with the NL phrase mapped to a program that captures the computation involving the pivoting entity (Appendix B).

Discussion Our SCFG uses idiomatic productions that capture domain-specific language expressions, together with simple domain-general rules (Herzig and Berant, 2019) to combine those idiomatic constructs to form compositional utterances. As we show in §4, both the base and idiomatic grammar sets are relatively compact, and we resort to strong paraphrasers to further "naturalize" synthetic utterances and bridge the language gap. In line with Su and Yan (2017) and Marzoev et al. (2020), we remark that such *functionality*-driven grammar engineering to cover representative patterns in real data using a small set of curated production rules is more efficient and cost-effective than example-driven annotation in classical supervised learning of semantic parsers, which requires labeling a sufficient number of parallel samples to effectively train a data-hungry neural model over a variety of underlying meanings and surface language styles.

Our approach is also orthogonal with the prior work Xu et al. (2020b), which uses large curated general-purpose grammars to attempt to model English syntax, while using weak domain-specific rules that are much easier to specify than our SCFG, but might not be as effective to capture idiomatic language patterns in the domain. On the other hand, grammar engineering can be potentially costly. Ideally, one could study representative samples from real data and come up with a small set of idiomatic productions in the above categories that are expressive enough for domains like GEO and SCHOLAR (§4). Still, the exact the amount of effort this process takes remains difficult to estimate. We present more discussion in §5.

3.1.2 Naturalness-driven Data Selection

To cover real programs in $\mathbb{D}_{\text{nat}}$ with complex structures while tackling the exponential sample space, we propose an efficient approach to sub-sample a small set of examples from this space as seed canonical data $\mathbb{D}_{\text{can}}$ (Fig. 1b) for paraphrasing. Our core idea is to only retain a set of examples $\langle \boldsymbol{u}, \boldsymbol{z} \rangle$ that most likely reflect the intents of real users. We use the probability $p_{\text{LM}}(\boldsymbol{u})$ measured by a language model to approximate the "naturalness" of canonical examples.[4] Specifically, given all canonical examples allowed by the grammar, we form buckets based on their derivation depth d. For each bucket $\mathbb{D}_{\text{can}}^{(d)}$, we compute $p_{\text{LM}}(\boldsymbol{u})$ for its examples, and group the examples using program templates as the key (*e.g.* $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ in Fig. 1 are grouped together). For each group, we find the example $\langle \boldsymbol{u}^*, \boldsymbol{z} \rangle$ with the highest $p_{\text{LM}}(\boldsymbol{u}^*)$, and discard other examples $\langle \boldsymbol{u}, \boldsymbol{z} \rangle$ if $\ln p_{\text{LM}}(\boldsymbol{u}^*) - \ln p_{\text{LM}}(\boldsymbol{u}) >$

[4] We use the GPT-2 XL model (Radford et al., 2019).

δ ($\delta = 5.0$), removing unlikely utterances from the group (*e.g.* $\boldsymbol{u}_1$).[5] Finally, we rank all groups in $\mathbb{D}_{\text{can}}^{(d)}$ based on $p_{\text{LM}}(\boldsymbol{u}^*)$, and retain examples in the top-K groups. This method offers trade-off between program coverage and efficiency and, more surprisingly, we show that using only $0.2\%\sim1\%$ top-ranked examples also results in significantly better final accuracy (§4).

3.2 Generating Validation Data

Zero-shot learning is non-trivial without a high-quality validation set, as the model might overfit on the (paraphrased) canonical data, which is subject to language and logical mismatch. While existing methods (Xu et al., 2020b) circumvent the issue using real validation data, in this work we create validation sets from paraphrased examples, making our method truly labeled data-free. Specifically, we consider a two-stage procedure. First, we run the iterative paraphrasing algorithm (§2) without validation, and then sample $\langle \boldsymbol{u}, \boldsymbol{z} \rangle$ from its output with a probability $p(\boldsymbol{u}, \boldsymbol{z}) \propto p_{\text{LM}}(\boldsymbol{u})^{\alpha}$ ($\alpha = 0.4$), ensuring the resulting sampled set $\mathbb{D}_{\text{par}}^{\text{val}}$ is representative. Second, we restart training using $\mathbb{D}_{\text{par}}^{\text{val}}$ for validation to find the best checkpoint. The paraphrase filtering model is also initialized with the parser trained in the first stage, which has higher precision and accepts more valid paraphrases. This is similar to iterative training of weakly-supervised semantic parsers (Dasigi et al., 2019), where the model searches for candidate programs for unlabeled utterances in multiple stages of learning.

4 Experiments

We evaluate our zero-shot parser on two datasets.

SCHOLAR (Iyer et al., 2017) is a corpus of user-issued queries to an academic database (Fig. 1). We use the version from HB19 with programs represented in λ-calculus logical forms. The sizes of the train/test splits are 577/211. Entities in utterances and programs (*e.g.* *Parsing paper in ACL*) are canonicalized to slots (*e.g.* `keyphrase0`, `venue0`), and are recovered before executing the programs. We found in the dataset by HB19, slots are paired with with random entities for execution (*e.g.* `keyphrase0`$\mapsto$*Optics*). Therefore reference programs are likely to execute to empty results, making metrics like answer accuracy more prone to false-positives. We fix all such examples in the dataset, as well as those with execution errors.

[5] δ chosen in pilot studies, similar to Zhang et al. (2019).

System	Supervision	SCHOLAR	GEO
Supervised†	Labeled Examples	79.7 ± 2.2	81.9 ± 5.3
OVERNIGHT†	Manual Paraphrases	41.0 ± 3.8	55.8 ± 6.4
GRANNO†	Real Utterances, Manual Paraphrase Detection	68.3 ± 1.6	69.4 ± 1.9
Our System	–	**75.5** ± 1.6	**74.1** ± 2.3

Table 1: Averaged denotation accuracy and standard deviation on TEST sets. Results are averaged with five random restarts. †Models originally from Herzig and Berant (2019) and run with five random restarts. Results from our model are tested *v.s.* GRANNO using permutation test with $p < 0.05$.

GEO (Zelle and Mooney, 1996) is a classical dataset with queries about U.S. geography (*e.g.* *Which rivers run through states bordering California?*). Its database contains basic geographical entities like cities, states, and rivers. We also use the release from HB19, of size 596/278.

Models and Configuration Our neural semantic parser uses a BERT$_{\text{Base}}$ encoder (Devlin et al., 2019) and an LSTM decoder with copy mechanism. The paraphraser is a BART$_{\text{Large}}$ model (Lewis et al., 2020). We use the same set of hyper-parameters for both datasets. Specifically, we synthesize canonical examples from the SCFG with a maximal program depth of 6, and collect the top-K ($K = 2{,}000$) GPT-scored sample groups for each depth as the seed canonical data $\mathbb{D}_{\text{can}}$ (§3.1.2), with two rounds of iterative paraphrasing and training (§2). The beam size for the paraphraser is 20. We create validation sets of size $2{,}000$ following §3.2. Refer to Appendix C for more details. Note that our model only uses the natural examples in both datasets for evaluation purposes, and the training and validation splits are *not* used during learning.

Measuring Language and Logical Gaps We measure the language mismatch between utterances in the paraphrased canonical ($\mathbb{D}_{\text{par}}$) and natural ($\mathbb{D}_{\text{nat}}$) data using **perplexities** of natural utterances in $\mathbb{D}_{\text{nat}}$ given by a GPT-2 LM fine-tuned on $\mathbb{D}_{\text{par}}$. For logical gap, we follow HB19 and compute the **coverage** of natural programs $\boldsymbol{z} \in \mathbb{D}_{\text{nat}}$ in $\mathbb{D}_{\text{par}}$.

Metric We report **denotation accuracy** on the execution results of predicted programs.[6] We ran all experiments with five random restarts and report the mean and standard deviation.

4.1 Results

In experiments, we first compare our model with existing approaches using labeled data. Next, we analyze how our proposed methods close the lan-

[6] We use SEMPRE (Berant et al., 2013) to execute λ-calculus logical forms in parallel.

guage and logical gaps. Tab. 1 reports test accuracies of various systems on the test sets, as well as their form of supervision. Specifically, the **supervised** parser uses a standard parallel corpus $\mathbb{D}_{\text{nat}}$ of real utterances annotated with programs. **OVERNIGHT** uses paraphrased synthetic examples $\mathbb{D}_{\text{par}}$ like our model, but with manually written paraphrases. **GRANNO** uses unlabeled real utterances $\boldsymbol{u}_{\text{nat}} \in \mathbb{D}_{\text{nat}}$, and manual paraphrase detection to pair $\boldsymbol{u}_{\text{nat}}$ with the canonical examples $\mathbb{D}_{\text{can}}$. Our model outperforms existing approaches without using any labeled data, while GRANNO, the currently most cost-effective approach, still spends \$155 in manual annotation (besides collecting real utterances) on the two datasets (Herzig and Berant (2019), HB19). This demonstrates that our zero-shot parser is a data-efficient and cost-effective paradigm to train semantic parsers for emerging domains. Still, our system falls behind supervised models trained on natural corpora $\mathbb{D}_{\text{nat}}$, due to language and logical gaps between $\mathbb{D}_{\text{par}}$ and $\mathbb{D}_{\text{nat}}$. Next, we explore whether our proposed methods are effective at narrowing the gaps and improving accuracy. Since the validation splits of the two datasets are small (< 100), we evaluate on the full training/validation splits (around 600 examples) to get more reliable results.

More expressive grammars narrow language and logical gaps We capture domain-specific language patterns using idiomatic productions to close language mismatch (§3.1.1). Tables 2 and 3 list the results when we gradually augment the grammar with different categories of idiomatic productions. More expressive grammars help close the language gap, as indicated by the decreasing perplexities. This is especially important for SCHOLAR, which has diverse NL expressions hard to infer from plain canonical utterances. For instance, it could be non-trivial to paraphrase canonical utterances with multi-hop (*e.g. Author that <u>cites</u> paper <u>by</u> X*) or superlative relations (*e.g. Topic of <u>the most number of</u> ACL paper*) to more idiomatic alternatives (*e.g.* "*Author that <u>cites</u> X*", and "*<u>The most popular topic for</u> ACL paper*"), while directly including such patterns in the grammar (+**Multihop Rel.** and +**Superlative**) is helpful. We also remark that the number of idiomatic productions we created is fairly compact (See Appendix B for a complete list).[7] We are able to

Grammar	Size	ACC.	PPL	Logical Cov. $\mathbb{D}_{\text{can}}$	$\mathbb{D}_{\text{par}}$
Base	46	66.3 ± 3.7	23.0	80.6	75.8
+Multihop Rel.[8]	11	67.0 ± 1.1	22.0	87.7	81.2
+Comparison	2	67.3 ± 2.4	21.7	86.5	80.2
+Superlative	13	77.8 ± 2.2	20.9	90.6	86.1
−Multihop Rel.	2	75.8 ± 3.4	20.8	83.9	81.1

Table 2: Ablation of grammar categories on SCHOLAR.

Grammar	Size	ACC.	PPL	Logical Cov. $\mathbb{D}_{\text{can}}$	$\mathbb{D}_{\text{par}}$
Base	68	64.5 ± 4.6	8.2	84.4	79.7
+Multihop Rel.	4	67.9 ± 4.0	8.1	83.6	79.7
+Superlative	9	72.8 ± 2.8	8.0	84.1	79.4
−Multihop Rel.	4	66.5 ± 3.7	8.2	84.1	80.0

Table 3: Ablation study of grammar categories on GEO.

improve the accuracy by 11% absolute with 26 rules on SCHOLAR, while achieving 8% gain using only 13 idiomatic productions on the simpler GEO domain with fewer entity types and relations.

Additionally, more expressive grammars also improve logical coverage. The last columns (**Logical Cov.**) of Tables 2 and 3 report the percentage of real programs that are covered by the seed canonical data before ($\mathbb{D}_{\text{can}}$) and after ($\mathbb{D}_{\text{par}}$) iterative paraphrasing. Intuitively, a single idiomatic production often captures compositional computations like multi-hop queries, allowing the grammar to generate more compositional programs under the same threshold on the derivation depth. Notably, with all the full grammar on SCHOLAR, the number of exhaustively generated examples with a depth of 6 is tripled ($530K \mapsto 1,700K$).

Moreover, recall that the seed canonical dataset $\mathbb{D}_{\text{can}}$ contains examples with highly-likely utterances under the LM (§3.1.2). Therefore, examples created by idiomatic productions are more likely to be included in $\mathbb{D}_{\text{can}}$. However, this could also be counter-productive, as such examples could dominate $\mathbb{D}_{\text{can}}$, "crowding out" other useful examples with lower LM scores. This likely explains the slightly decreased logical coverage on GEO (Tab. 3), as more than 30% samples in the LM filtered $\mathbb{D}_{\text{can}}$ include idiomatic multi-hop relations directly connecting geographic entities with their countries (*e.g.* "*City <u>in</u> US*"), while such examples only account for $\sim 8\%$ of real data. While the over-representation issue might not negatively impact accuracy, we leave generating more balanced synthetic data as important future work.

[7]The base grammar is adapted from HB19, which defines entity types, example entities and (synonyms of) relations in each domain.

[8]This category only considers merging relation chains, and does not include superlative rules involving multiple relations.

	K	TRAIN Size $\lvert\mathbb{D}_{\text{can}}\rvert$	$\lvert\mathbb{D}_{\text{par}}\rvert$	ACC	PPL	Logical Coverage $\mathbb{D}_{\text{can}}$	$\mathbb{D}_{\text{par}}$
SCHOLAR	500	1.6K	45K	74.0 ±3.7	22.0	79.4	76.0 (14.5)
	1,000	3.1K	80K	75.9 ±1.7	21.4	88.0	82.0 (9.4)
	2,000	5.5K	130K	77.8 ±2.2	20.9	90.6	86.1 (7.5)
	4,000	9.2K	202K	78.4 ±1.7	20.7	91.9	87.4 (4.9)
	8,000	17K	331K	75.5 ±2.1	21.5	92.0	88.2 (2.9)
GEO	500	1.4K	30K	61.6 ±5.4	8.4	70.3	64.4 (14.2)
	1,000	2.6K	55K	68.5 ±7.7	8.2	80.5	74.9 (9.0)
	2,000	5.4K	113K	72.8 ±2.8	8.0	84.1	79.4 (5.2)
	4,000	11K	183K	67.5 ±6.3	8.2	84.9	78.3 (3.1)
	8,000	16K	243K	67.9 ±4.5	8.2	85.4	78.0 (2.1)

Table 4: Results on SCHOLAR and GEO with varying amount of canonical examples in the seed training data.

Finally, we note that the logical coverage drops after paraphrasing ($\mathbb{D}_{\text{can}}$ *v.s.* $\mathbb{D}_{\text{par}}$ in Tables 2 and 3). This is because for some samples in $\mathbb{D}_{\text{can}}$, the paraphrase filtering model rejects all their paraphrases. We provide further analysis later in §5.

Do smaller logical gaps entail better performance? As in §3.1.2, to make learning tractable in face of the exponential space of canonical samples, the seed canonical data $\mathbb{D}_{\text{can}}$ used in iterative paraphrasing only consists of top-K highest-scoring examples under GPT-2 for each program depth. However, using a smaller cutoff threshold K might sacrifice logical coverage, as fewer examples are in $\mathbb{D}_{\text{can}}$. To investigate this trade-off, we report results with varying K in Tab. 12. Notably, with $K = 1,000$ and around $3K$ seed canonical data $\mathbb{D}_{\text{can}}$ (before iterative paraphrasing), $\mathbb{D}_{\text{can}}$ already covers 88% and 80% natural programs on SCHOLAR and GEO, resp. This small portion of samples only account for 0.2% (1%) of the full set of $1.7M+$ ($0.27M$) canonical examples exhaustively generated from the grammar on SCHOLAR (GEO). This demonstrates our data selection approach is effective in maintaining learning efficiency while closing the logical gap.

More interestingly, while larger K further closes the logical gap, the accuracy might not improve accordingly. This is possibly because while the coverage of real programs increases, the percentage of such programs in paraphrased canonical data $\mathbb{D}_{\text{par}}$ (numbers in parentheses) actually drops. Out of the remaining 90%+ samples in $\mathbb{D}_{\text{par}}$ not covered in $\mathbb{D}_{\text{nat}}$, many have unnatural intents that real users are unlikely to issue (*e.g.* "*Number of titles of papers with the smallest citations*", or "*Mountain whose elevation is the length of Colorado River*"). Such *unlikely* samples are potentially harmful to the model, causing worse language mismatch, as suggested by the increasing perplexity when $K = 8,000$.

Model	SCHOLAR	GEO
Full Model (Tab. 4, K=2000)	77.8 ±2.2	72.8 ±2.8
Baselines for Selecting Canonical Samples		
∟ No GPT-2 scoring (Random)	69.7 ±9.0	65.5 ±4.7
∟ No balancing program depths	63.0 ±2.0	46.5 ±7.1
Baseline for Creating Validation Data		
∟ Random split of $\mathbb{D}_{\text{can}}$	74.1 ±1.5	69.7 ±3.3

Table 5: Comparing our model with baseline approaches for selecting canonical samples and generating validation data.

Paraphraser	SCHOLAR Tok. F_1↓	τ↓	ACC.↑	GEO Tok. F_1↓	τ↓	ACC.↑
Ours	70.3	0.71	77.8	69.2	0.78	72.8
Xu et al. (2020b)	72.4	0.94	69.9	74.5	0.95	62.3

Table 6: Systems with different paraphrasers. We report end-to-end denotation accuracy, as well as F_1 and Kendall's τ rank coefficient between utterances and their paraphrases.

Similar to HB19, we observe around one-third of samples in $\mathbb{D}_{\text{can}}$ and $\mathbb{D}_{\text{par}}$ are unlikely. As we later discuss in §5, such unlikely utterances often have noisier paraphrases, which hurts the quality of $\mathbb{D}_{\text{par}}$.

Comparing Data Selection Methods Next, we compare our proposed canonical data selection approach using GPT-2 with several baselines (Tab. 5 *Upper Half*). First, randomly choosing examples from each level of program depth instead of using the top-K GPT-scored ones results is less effective with higher variance. Further simplifying the procedure without constraining on equal sample size across program depths leads to significantly worse results, due to the scarcity of likely examples with simpler programs in the resulting sample set.

Impact of Validation Data We generate validation data from samples of the paraphrased data in an initial run (§3.2). Tab. 5 (*Lower Half*) compares this strategy with a baseline approach, which randomly splits the seed canonical examples in $\mathbb{D}_{\text{can}}$ into training and validation sets, and runs the iterative paraphrasing algorithm on the two sets in parallel, with paraphrases from both sets filtered by the filtering model. This approach underperforms, since some canonical samples with program patterns in the natural data $\mathbb{D}_{\text{nat}}$ can be partitioned into the validation split, and not used for training.

Impact of Paraphrasers We rely on strong paraphrasers to generate diverse utterances to close the language gap. Tab. 6 compares the system using our paraphraser and the one in Xu et al. (2020b). Both are based on BART, while ours is fine-tuned to encourage lexically and syntactically diverse outputs (Appendix A). We measure lexical diversity using token-level F_1 between the original and para-

Example 1 (Uncommon Concept)		
$\boldsymbol{u}_1$	*Venue of paper by* $\texttt{author}_0$ *and published in* $\texttt{year}_0$	
$\boldsymbol{u}'_{1,1}$	$\texttt{author}_0$*'s paper, published in* $\texttt{year}_0$	✘
$\boldsymbol{u}'_{1,2}$	*Where the paper was published by* $\texttt{author}_0$ *in* $\texttt{year}_0$*?*	✘
$\boldsymbol{u}'_{1,3}$	*Where the paper was published in* $\texttt{year}_0$ *by* $\texttt{author}_0$*?*	✘
$\boldsymbol{u}^*_{\text{nat}}$	*Where did* $\texttt{author}_0$ *publish in* $\texttt{year}_0$*?*	(Wrong Answer)
Example 2 (Novel Language Pattern)		
$\boldsymbol{u}_2$	*Author of paper published in* $\texttt{venue}_0$ *and in* $\texttt{year}_0$	
$\boldsymbol{u}'_{2,1}$	*Author of papers published in* $\texttt{venue}_0$ *in* $\texttt{year}_0$	✔
$\boldsymbol{u}'_{2,2}$	*Who wrote a paper for* $\texttt{venue}_0$ *in* $\texttt{year}_0$	✘
$\boldsymbol{u}'_{2,3}$	*Who wrote the* $\texttt{venue}_0$ *paper in* $\texttt{year}_0$	✘
$\boldsymbol{u}^*_{\text{nat}}$	$\texttt{venue}_0$ $\texttt{year}_0$ *authors*	(Correct)
Example 3 (Unnatural Canonical Utterance)		
$\boldsymbol{u}_3$	*Author of paper by* $\texttt{author}_0$	
$\boldsymbol{u}'_{3,1}$	*Author of the paper written by* $\texttt{author}_0$	✔
$\boldsymbol{u}'_{3,2}$	*Author of* $\texttt{author}_0$*'s paper*	✔
$\boldsymbol{u}'_{3,3}$	*Who wrote the paper* $\texttt{author}_0$ *wrote?*	✘
$\boldsymbol{u}^*_{\text{nat}}$	*Co-authors of* $\texttt{author}_0$	(Wrong Answer)
Example 4 (Unlikely Example)		
$\boldsymbol{u}_4$	*Paper in* $\texttt{year}_0$ *and whose author is not the most cited author*	
$\boldsymbol{u}'_{4,1}$	*A paper published in* $\texttt{year}_0$ *that isn't the most cited author*	✔
$\boldsymbol{u}'_{4,2}$	*What's not the most cited author in* $\texttt{year}_0$	✔
$\boldsymbol{u}'_{4,3}$	*In* $\texttt{year}_0$*, he was not the most cited author*	✔

Table 7: Case Study on SCHOLAR. We show the seed canonical utterance $\boldsymbol{u}_i$, the paraphrases $\boldsymbol{u}'_{i,j}$, and the relevant natural examples $\boldsymbol{u}^*_{\text{nat}}$. ✔ and ✘ denote the correctness of paraphrases. ✘ denotes false negatives of the filtering model (correct paraphrases that are filtered), ✔ denotes false positives (incorrect paraphrases that are accepted). Entities are canonicalized with indexed

phrased utterances $\langle \boldsymbol{u}, \boldsymbol{u}' \rangle$ (Rajpurkar et al., 2016; Krishna et al., 2020). For syntactic divergence, we use Kendall's τ (Lapata, 2006) to compute the ordinal correlation of $\boldsymbol{u}$ and $\boldsymbol{u}'$. Our paraphraser outputs more diverse paraphrases (*e.g. What is the biggest state in US?*) from the source (*e.g. State in US and that has the largest area*), as indicated by lower token-level overlaps and ordinal coefficients, comparing to the existing paraphraser (*e.g. The state in US with the largest surface area*). Still, our paraphraser is not perfect, as discussed next.

5 Limitations and Discussion

Our parser still lags behind the fully supervised model (Tab. 1). To understand the remaining bottlenecks, we show representative examples in Tab. 7.

Low Recall of Filter Model First, the recall of the paraphrase filtering model is low. The filtering model uses the parser trained on the paraphrased data generated in previous iterations. Since this model is less accurate, it can incorrectly reject valid paraphrases $\boldsymbol{u}'$ (✘ in Tab. 7), especially when $\boldsymbol{u}'$ uses a different sentence type (*e.g.* questions) than the source (*e.g.* statements). Empirically, we found the recall of the filtering model at the first iteration of the second-stage training (§3.2) is only around 60%. This creates logical gaps, as paraphrases of examples in the seed canonical data $\mathbb{D}_{\text{can}}$ could be rejected by the conservative filtering model, leaving no samples with the same programs in $\mathbb{D}_{\text{par}}$.

Imperfect Paraphraser The imperfect paraphraser could generate semantically incorrect predictions (*e.g.* $\boldsymbol{u}'_{1,1}$), especially when the source canonical utterance contains uncommon or polysemic concepts (*e.g. venue* in $\boldsymbol{u}_1$), which tend to be ignored or interpreted as other entities (*e.g. sites*). Besides rare concepts, the paraphraser could also fail to rewrite canonical utterances using more idiomatic syntax, like changing the mentioning of a conference using prepositional phrases ($\boldsymbol{u}_2$) to compound nouns ($\boldsymbol{u}^*_{\text{nat}}$ in Example 2). While the model might still correctly answer $\boldsymbol{u}^*_{\text{nat}}$, $\boldsymbol{u}^*_{\text{nat}}$'s perplexity is high, suggesting language mismatch.

Unnatural Canonical Utterances While we have attempted to close the language gap by generating more idiomatic canonical utterances, some of them are still not natural enough for the paraphraser to rewrite. This is especially problematic for relations not covered by our idiomatic productions, such as the co-authorship relation in Example 3. While this issue could be mitigated using additional production rules, grammar engineering could still remain challenging, as elaborated later.

Unlikely Examples Besides the unnatural canonical utterances with clunky surface expressions but are still logically plausible, $\mathbb{D}_{\text{can}}$ also contains around 30% unlikely examples with both unnatural utterances and convoluted meanings that almost certainly will not appear in real data (*e.g.* $\boldsymbol{u}_4$). Similar to unnatural utterances, their paraphrases are also much noisier (*e.g.* $\boldsymbol{u}'_{4,*}$), with only around 30% paraphrasing accuracy, compared to 70% for the likely ones. The filtering model is also less effective on unlikely examples (false positives ✔). These noisy samples will eventually hurt performance of the parser. We leave modeling utterance naturalness as important future work.

Cost of Grammar Engineering Our approach relies on an expressive SCFG to bridge the language and logical gaps between synthetic and real data. While we have attempted to standardize the process of grammar construction by designing idiomatic productions following a set of representative grammar categories, grammar engineering still remains a non-trivial task. One need to have a good sense of the idiomatic language patterns that would frequently appear in real-world data, which requires performing user study or access to sampled data. Encoding those language patterns as production rules could also take a reasonable amount of time, depending on various factors, such as the complexity of the target domain and the proficiency of the user in the grammar formalism (λ-calculus) used by our system.

Still, we remark that most of the curated productions have simple syntactic constructs (a single verb, preposition, or adjective phrase, more in Appendix B.2.2), and we are able to significantly improve the performance over the base grammar (Tables 2 and 3) using a relatively compact idiomatic grammar ($10 \sim 30$ rules on two datasets). Additionally, considering that the size of those idiomatic rules is orders of magnitude smaller than the size of the annotated parallel examples in the original datasets (around 800), it is safe to assume that for users familiar with the grammar formalism, curating such a small set of grammar rules for domains similar to SCHOLAR and GEO is more efficient than labeling parallel samples in the original datasets. For the latter task the user would have to consider other factors, such as the coverage of compositional logical form patterns and language expressions, while our system automatically synthesizes compositional samples with diverse language style by composing (idiomatic) productions and iterative paraphrasing. Moreover, the paraphrased canonical examples synthesized from a compact curated grammar could also be used to bootstrap the collection of high-quality parallel data. Finally, creation of grammar rules could potentially be simplified by defining them using natural language instead of logical forms, reminiscent of studies on naturalizing programs using canonical language (Wang et al., 2017; Shin et al., 2021; Herzig et al., 2021).

6 Related Work

To mitigate the paucity of labeled data, the field has explored various supervision signals. Specifically, **weakly-supervised** methods leverage the denotations of utterances as indirect supervision (Clarke et al., 2010; Krishnamurthy and Mitchell, 2012), with programs modeled as latent variables (Berant et al., 2013; Pasupat and Liang, 2015). Optimization is challenging due to the noisy binary reward of execution correctness (Agarwal et al., 2019), calling for better learning objectives (Guu et al., 2017; Wang et al., 2021a) or efficient search algorithms for latent programs (Krishnamurthy et al., 2017; Liang et al., 2017, 2018; Muhlgay et al., 2019). Next, **semi-supervised** models leverage extra unlabeled utterances, using techniques like self-training (Konstas et al., 2017) or generative models (Kociský et al., 2016; Yin et al., 2018). As a step further, **unsupervised** methods only use unlabeled utterances (Cao et al., 2019), and leverage linguistic scaffolds (*e.g.* dependency trees) to infer programs with similar structures (Poon, 2013). Like our model, such methods use lexicons to capture alignments between NL phrases and logical predicates (Goldwasser et al., 2011), while our method does not require real utterances. Finally, methods based on OVERNIGHT (Wang et al., 2015) synthesize parallel corpora from SCFGs (Cheng et al., 2019; Xu et al., 2020a) or neural sequence models (Guo et al., 2018), and attempt to bridge the gaps between canonical and real utterances via paraphrase detection (Herzig and Berant, 2019) and generation (Su and Yan, 2017; Shin et al., 2021; Wu et al., 2021), or representation learning (Marzoev et al., 2020).

7 Conclusion and Future Work

In this paper, we propose a zero-shot semantic parser that closes the language and logical gaps between synthetic and real data. on SCHOLAR and GEO, our system outperforms other annotation-efficient approaches with zero labeled data.

There are several import avenues for future work. First, dedicated approaches to generate syntactically diverse paraphrases using latent variable models, such as Hosking and Lapata (2021) and Hosking et al. (2022), could potentially improve performance. Additionally, systematic comparison with AUTOQA could help elucidate the impact of grammar quality to zero-shot semantic parsing, although this was not covered in this study due to complexities in porting λ-calculus logical forms to the specialized formalism in AUTOQA. Next, careful human studies to understand the amount of efforts required for grammar engineering would provide more insights on the practicality of our approach. Finally, generalizing our approach to domains with more complex schemas (*e.g.* ATIS) is an important direction, which traditionally relies on careful feature engineering to reduce the amount of annotated data (Poon, 2013).

Acknowledgements

We are grateful to Jonathan Herzig for sharing the code for GRANNO, and Silei Xu and Giovanni Campagna for questions regarding AUTOQA. Pengcheng Yin was supported in part by an IBM Ph.D. fellowship. We thank our anonymous reviewers for their insightful comments.

References

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *ICML*.

Ekin Akyürek, Afra Feyza Akyurek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. In *Proceedings of ICLR*.

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8.

Thomas W. Anthony, Zheng Tian, and David Barber. 2017. Thinking fast and slow with deep learning and tree search. In *NIPS*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transaction of ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of ACL*.

Jianpeng Cheng, Siva Reddy, V. Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45:59–94.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CoNLL*.

Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke S. Zettlemoyer, and Eduard H. Hovy. 2019. Iterative search for weakly supervised semantic parsing. In *Proceedings of NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

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 ACL*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Proceedings of NeurIPS*.

Dan Goldwasser, Roi Reichart, J. Clarke, and D. Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of ACL*.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and M. Zhou. 2018. Question generation from sql queries improves neural semantic parsing. In *Proceedings of EMNLP*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of EMNLP*.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of ACL*.

Jonathan Herzig and Jonathan Berant. 2019. Don't paraphrase, detect! rapid and effective data collection for semantic parsing. In *Proceedings of EMNLP*.

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*, abs/2104.07478.

Tom Hosking and Mirella Lapata. 2021. Factorising meaning and form for intent-preserving paraphrasing. In *Proceedings of ACL*.

Tom Hosking, Hao Tang, and Mirella Lapata. 2022. Hierarchical sketch induction for paraphrase generation. *ArXiv*, abs/2203.03463.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of ACL*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Tomás Kociský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of EMNLP*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of ACL*.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of EMNLP*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of EMNLP*.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of EMNLP*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of ICML*.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

M. Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, A. Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL*.

Fei Li and H. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, 8:73–84.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Proceedings of NIPS*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Alana Marzoev, S. Madden, M. Kaashoek, Michael J. Cafarella, and Jacob Andreas. 2020. Unnatural language processing: Bridging the gap between synthetic and natural language data. *ArXiv*, abs/2004.13645.

Dor Muhlgay, Jonathan Herzig, and Jonathan Berant. 2019. Value-based search in execution space for mapping instructions to programs. In *Proceedings of NAACL*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of ACL*.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL*.

Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Y. Altun. 2019. Generating logical forms from graph representations of text and entities. In *Proceedings of ACL*.

Richard Shin, C. H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, D. Klein, J. Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. *ArXiv*, abs/2104.08768.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of EMNLP*.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021a. Learning from executions for semantic parsing. In *Proceedings of NAACL*.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021b. Learning to synthesize data for semantic parsing. In *Proceedings of NAACL*.

Sida I. Wang, Samuel Ginn, Percy Liang, and Christopher D. Manning. 2017. Naturalizing a programming language via interactive learning. In *ACL*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of ACL*.

J. Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of ACL*.

Shan Wu, Bo Chen, Chunlei Xin, Xianpei Han, Le Sun, Weipeng Zhang, Jiansong Chen, Fan Yang, and Xunliang Cai. 2021. From paraphrasing to semantic parsing: Unsupervised semantic parsing via synchronous semantic decoding. In *Proceedings of ACL*.

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 *Proceedings of CIKM*.

Silei Xu, Sina J. Semnani, Giovanni Campagna, and M. Lam. 2020b. Autoqa: From databases to qa semantic parsers with only synthetic training data. In *Proceedings of EMNLP*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of ACL*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of ACL*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of EMNLP*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI*.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase Adversaries from Word Scrambling. In *NAACL*.

On The Ingredients of an Effective Zero-shot Semantic Parser
Supplementary Materials

A Paraphraser

Central to our approach is a paraphrase generation model $p(\boldsymbol{u} \mapsto \boldsymbol{u}')$, which paraphrases a canonical utterance $\boldsymbol{u}$ to an alternative sentence $\boldsymbol{u}'$ that is possibly more natural and linguistically diverse. To improve the diversity of generated paraphrases, we paraphrase $\boldsymbol{u}$ to multiple candidate rewrites $\{\boldsymbol{u}'\}$ using beam search. We tested multiple strategies (*e.g.* nucleus sampling) to improve diversity of paraphrases via ensuring quality, and found beam search yields the best end-to-end performance.

To generate high-quality paraphrases for open-domain utterances, we parameterize $p(\boldsymbol{u} \mapsto \boldsymbol{u}')$ using generative pre-trained LMs (BART$_{\text{Large}}$).[9] The LM is fine-tuned on a corpus of $70K$ high-quality paraphrases sub-sampled from PARANMT (Wieting and Gimpel, 2018) released by Krishna et al. (2020), where samples are carefully constructed to ensure the lexical and syntactical diversity of target paraphrases. To further improve the syntactic diversity of paraphrases from statement-style inputs (*e.g.* $\boldsymbol{u}_2$, Fig. 1), we apply force decoding with WH-prefixes (*e.g. What*, *When*, *How many*, based on the answer type) to half hypotheses in the beam to generate question paraphrases (*e.g.* paraphrases prefixed with *"How many"* for $\boldsymbol{u}_3$ in Fig. 1).

Filtering Paraphrases While our paraphraser is strong, it is still far from perfect, especially when tasked with paraphrasing utterances found in arbitrary down-stream domains. For example, two ambiguous utterances "*Author that cites A*" and "*Author cited by A*" could get the same paraphrase "*Who cites A?*". Such noisy paraphrases will bring noise to learning and hurt performance. To filter potentially incorrect outputs, we follow Xu et al. (2020b) and use the parser trained on the paraphrased data generated in the preceding iteration (or the seed canonical data at the beginning of training) to parse each paraphrased utterance, and only retain those for which the parser could successfully predict its program. Admittedly such a stringent criterion will sacrifice recall, but empirically we found it works well. We present more analysis in the case study in §4.

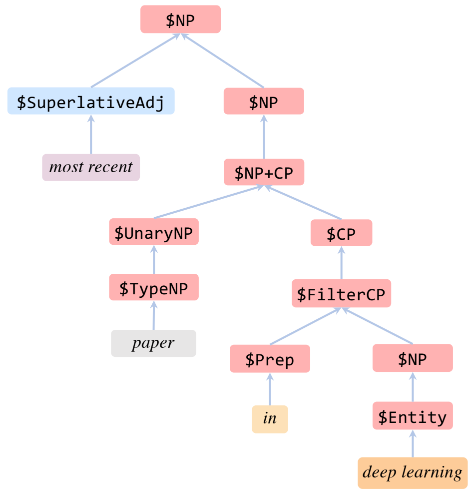
B Synchronous Grammar

Our synchronous grammar is adapted from Herzig and Berant (2019) and Wang et al. (2015), which specifies alignments between NL expressions and logical form constituents in λ-calculus s-expressions.[10] The grammar consists of a set of domain-general production rules (Appendix B.1), plus domain-specific rules specifying domain lexicons (Appendix B.2.1) and idiomatic productions (Appendix B.2.2). Specifically, domain-general productions define (1) generic logical operations like `count` and `superlative` (*e.g.* r_3, Fig. 1), and (2) compositional rules to construct utterances following English syntax (*e.g.* r_1, Fig. 1). Domain-specific rules, on the other hand, are typically used to define task-dependent lexicons like types (*e.g.* `author`), entities (*e.g.* `alan_turing`), and relations (*e.g.* `citations`) in the database. This work also introduces idiomatic productions to specific common NL expression catered to a domain, as outlined in §3.1.1 and detailed later.

B.1 Domain-General Grammar

Tab. 8 lists example domain-general productions in our SCFG. Fig. 2 shows the derivation that applies those productions to generate an example utterance and program. Each production has a syntactic body, specifying how lower-level syntactic constructs are composed to form more compositional utterances, as well as a *semantic function*, which defines how programs of child nodes are composed to generate a new program. For instance, the production r_3 in Tab. 8 generates a noun phrase from a unary noun phrase `UnaryNP` (*e.g. paper*) and a complementary phrase `CP` (*e.g. in deep learning*) by concatenating the child nodes `UnaryNP` and `CP` (*e.g. paper in deep learning*). On the program side, the programs of two child nodes on Fig. 2 are:

[9] We use the official implementation in `fairseq`, https://github.com/pytorch/fairseq.
[10] We use the implementation in `Sempre`, https://github.com/percyliang/sempre



Most recent paper in deep learning

```
(
  call listValue (
     call superlative
       (
         call filter
         (
            call getProperty
            (call singleton fb:en.paper)
            (string ! type)
         )
         (string paper.keyphrase)
         (string =)
         fb:en.keyphrase.deep_learning
       )
     (string max)
     (string paper.publication_year)
   )
)
```

Figure 2: (a) The derivation tree (production rule applications) to generate the example utterance and its program. (b) The program defined in s-expression.

```
# Get all entities whose type is paper
$UnaryNP: call getProperty (call singleton fb:en.paper) (string !type)
# A lambda function that returns entities in x whose relation paper.keyphrase is deep_learning
$CP: lambda x (call
          filter (x)
          (string paper.keyphrase)
          (string =)
          (fb:en.keyphrase.deep_learning))
```

where the program of UnaryNP is an entity set of papers, and the program of NP is a lambda function with a variable x, which filters the entity set. The semantic function of r_3 specifies how these two programs should be composed to form the program of their parent node NP+CP, which performs β reduction, assigning the entity set returned by UnaryNP to the variable x:

```
# Get all papers whose keyphrase is deep learning
$NP+CP: (call
          filter (
            call getProperty (call singleton fb:en.paper) (string !type)
          )
          (string paper.keyphrase)
          (string =)
          (fb:en.keyphrase.deep_learning))
```

B.2 Domain-specific Grammar

B.2.1 Lexicons and Base Productions

The domain-specific grammar uses a set of base productions to define the task-dependent lexicon, which specifies the mapping between database elements and their natural language expressions. There are three types of elements in the database of a domain: entity types (*e.g.* author), entities (*e.g.* alan_turing) and relations (*e.g.* author.paper), each associated with a base production rule to map them into NL phrases (*e.g.* "*author*", "*Alan Turing*", and "*writes*"). A DB element (*e.g.* the entity type keyphrase) could also have multiple base productions describing their synonyms (*e.g.* "*keyphrase*" and "*topic*"). The base lexicon in our system is adapted from GRANNO (Herzig and Berant, 2019)[11].

[11] https://github.com/jonathanherzig/semantic-parsing-annotation/tree/master/grammars

Id	Productions (Syntactic Body and Semantic Function)	Description
r_1	`NP↦SuperlativeAdj NP`	*e.g. most recent* $\boxed{?}$
	`lambda rel, sub (` `  call superlative (var sub) (string max) (var rel))`	lambda function to get the subject `sub` with the largest relation `rel`
r_2	`NP↦NP+CP`	A noun phrase head `NP` and a complementary phrase body `CP` (*e.g. paper in deep learning*)
	`IdentityFn`	An identity function returning child program
r_3	`NP+CP↦UnaryNP CP`	*e.g. paper in deep learning*
	`Lambda Beta Reduction: f(var x)`	Perform beta reduction, applying the function from `CP` (*e.g. in deep learning*) to the value of `UnaryNP` (*e.g. paper*)
r_4	`UnaryNP↦TypeNP CP` `IdentityFn`	Entity types, *e.g. paper*
r_5	`CP↦FilterCP` `IdentityFn`	—
r_6	`FilterCP↦Prep NP`	*e.g. in deep learning*
	`lambda rel, obj, sub (` `  call filter (var sub) (var rel) (string =) (var obj))`	Create a lambda function, which filters entities in a list `sub` such that its relation `rel` (*e.g. topic*) equals `obj` (*e.g. deep learning*)
r_5	`NP↦Entity` `IdentityFn`	Entity noun phrases *e.g. deep learning*

Table 8: Example domain-general productions rules in the SCFG

B.2.2 Idiomatic Productions

Here we describe the two categories of idiomatic productions. Readers are referred to Tables 9 and 11 for the list of productions used in SCHOLAR and GEO.

Multi-hop Relations We create idiomatic productions for non-compositional NL phrases of multi-hop relations (*e.g. Author that writes paper in ACL*). We augment the database with entries for those multi-hop relations (*e.g.* ⟨`X`, `author.publish_in`, `acl`⟩), and then create productions in the grammar aligning those relations with their NL phrases (*e.g.* r_1 in Tab. 9).

Note that the productions defining different relations might have the same syntactic body, for example, r_1 and r_2 in Tab. 9. Since our synthesis algorithm based on Herzig and Berant (2019) performs type checking before composing productions, when it generates an utterance like *Author that publish in ACL*, only r_1 will be used, because the entity span *ACL* has a type (conference) `venue`, not `journal`.

Comparatives and Superlatives We also create productions for idiomatic comparatives and superlative expressions. Those productions specify the NL expressions for the comparative/superlative form of some relations. For example, for the relation `paper.publication_year` with objects of date time, its superlative form would be *most recent* and *first* (r_{14} in in Tab. 9), while its comparative form could be prepositional phrases like *published before* (r_{12}) and *published after*. Those productions define the lexicons for comparative/superlative expressions, and could be used by the domain-general rules like r_1 in Tab. 8 to compose utterances (*e.g.* Fig. 2).

Besides superlative expressions for relations whose objects are measurable, we also create idiomatic expressions for relations with countable subjects or objects. As an example, the utterance "*The most popular topic for papers in ACL*" involves grouping ACL papers by topic and return the most frequent one. Such computation is captured by the `CountSuperlative` operation in our SCFG based on Wang et al. (2015), and we create productions aligning those relations with the idiomatic noun phrases describing their superlative form (*e.g.* r_{16} in Tab. 9).

Perhaps the most interesting form of superlative relations are those involving reasoning with additional entities. For instance, the relation in *"venue that X publish mostly in"* between the entity `author` and `venue` implicitly involves counting the `papers` that the author *X* publishes. For those relations, we create "macro" productions (*e.g.* r_{20} in Tab. 10), which defines the lambda function that computes the answer (*e.g.* return the publication venue where *X* publishes the most number of papers) given the arguments

Id	Production Body (LHS↦RHS and Semantic Function)	Description
	Multi-hop Relations	
	Entity Tyle: `Author`	
r_1	`RelVP`↦*publish in* `ConstantFn (string author.publish_in)`	Verb phrase for multi-hop relation *author that writes paper in ACL*
r_2	`RelVP`↦*publish in* `ConstantFn (string author.publish_in_journal)`	Similar relation for journal publications.
r_3	`RelNP`↦*keyphrase | topic* `ConstantFn (string author.keyphrase)`	Noun phrase for multi-hop relation *keyphrase|topic of Alan Turing*, chaining the two relations *topic of paper by Alan Turing*
r_4	`RelVP`↦*works on* `ConstantFn (string author.keyphrase)`	Verb phrase for the same relation
r_5	`RelVP`↦*cites* `ConstantFn (string author.cites)`	Verb phrase ofr the multi-hop relation *author who cites Alan Turing*, chaining the three relations: *author of paper that cites paper by Alan Turing*
r_6	`RelVP`↦*cites* `ConstantFn (string author.cites_paper)`	Verb phrase chaining the relations *author of paper that cites paper_name*
	Entity Type: `Paper`	
r_7	`RelVP`↦*cites* `ConstantFn (string paper.cites_author)`	Verb phrase chaining the relations *Paper that cites paper by Alan Turing*
	Entity Type: `Venue`	
r_8	`RelNP`↦*topic* `ConstantFn (string venue.keyphrase)`	Noun phrase in multi-hop relation *topic of ACL*, chaining the two relations *topic of paper published in ACL*
r_9	`Prep`↦*in* `ConstantFn (string venue.keyphrase)`	Prepositional phrase describing the same relation (*e.g. Venue in NLP*)
	Entity Type: `Journal`	
r_{10}	`RelNP`↦*topic* `ConstantFn (string venue.keyphrase)`	Noun phrase in multi-hop relation *topic of Nature*, similar to the one for venues.
r_{11}	`Prep`↦*in* `ConstantFn (string venue.keyphrase)`	Prepositional phrase describing the same relation
	Comparative Relations	
r_{12}	`ComparativeLtPREP`↦*published before* `ConstantFn (string paper.publication_year)`	Comparative prepositions to describe publication dates
r_{13}	`ComparativeGtPREP`↦*published after* `ConstantFn (string paper.publication_year)`	Comparative prepositions to describe publication dates
	Superlative Relations	
	Entity Type: `Paper`	
r_{14}	`SuperlativeAdj`↦*most recent | first* `ConstantFn (string paper.publication_year)`	Superlative adjectives to describe publication dates
r_{15}	`SuperlativeMinAdj`↦*most cited* `ConstantFn (string paper.citation_count)`	Superlative adjectives to describe the a paper's citations
	Entity Type: `Keyphrase`	
r_{16}	`CountSuperlativeNP`↦*the most popular topic for* `ConstantFn (string keyphrase.paper)`	Superlative form to refer to the most frequent keyphrase for a set of paper, *e.g. the most popular topic for paper in ACL*
	Entity Type: `Venue`	
r_{17}	`MultihopCountSuperlativeNP`↦*the most popular venue for* `ConstantFn (string venue.keyphrase)`	Superlative form to refer to the most popular venue for paper about a topic, *e.g. the most popular venue for paper in deep learning*
	Entity Type: `Dataset`	
r_{18}	`MultihopCountSuperlativeNP`↦*the most popular dataset for* `ConstantFn (string dataset.paper)`	Superlative form to refer to the most popular dataset used by a set of paper

Table 9: List of example idiomatic productions used in SCHOLAR (to be continued in Table 10).

(*e.g.* an author *X*).

Id	Production Body (LHS↦RHS and Semantic Function)	Description
	Superlative Relations (cont'd)	
	Entity Tyle: `Author`	
r_{19}	`SuperlativeMinAdj`↦*most cited* `ConstantFn (string author.citation_count)`	Superlative adjectives to describe the an authors citations, *e.g.*, *most cited author in deep learning*

r_{20}	`MacroVP`↦*publish mostly in*
	`lambda author, venue (`
	`  call countSuperlative`
	`    (var venue)`
	`    (string max)`
	`    (string venue.paper)`
	`    (call getProperty (var author) (string author.paper))`
	`  )`
		Superlative form of the verb relational phrase *Author that publish mostly in ACL* with complex computation. `countSuperlative` returns the entity `x` in `venue` for which the papers in `x` (via relation `venue.paper`) has the largest intersection with papers by `author` (via realtion `author.paper`)

r_{21}	`MacroVP`↦*publish mostly in* `lambda author, journal ...`	Similar relation for journals.
r_{22}	`MacroVP`↦*publish mostly in* `lambda author, keyphrase ...`	Similar relation for topics of paper.
r_{23}	`MacroVP`↦*cites* `$NP` *the most* `lambda author, author ...`	*e.g. Author that cites Alan Turing the most.*
r_{24}	`MacroVP`↦*cites* `$NP` *the most* `lambda author, paper ...`	*e.g. Author that cites semantic parsing paper the most.*
r_{25}	`MacroVP`↦*cites the most* `lambda author, paper ...`	Similar expression in reversed form.*e.g. Author that semantic parsing paper cites the most.*
r_{26}	`MacroNPPrep`↦*The most productive author for* `lambda author, paper ...`	*e.g. The most productive author for paper in deep learning*

Table 10: List of example idiomatic productions used in SCHOLAR (continued). Semantic functions are simplified for illustration purpose. Refer to https://github.com/percyliang/sempre/blob/master/TUTORIAL.md for more details on λ-calculus SCFGs.

Id	Production Body (LHS↦RHS and Semantic Function)	Description
	Multi-hop Relations	
r_1	`RelPrep`↦*in* `ConstantFn (string city.located_in_country)`	Prepositional phrase for multi-hop relation *cities in the US*, chaining the two relations *city in state in the US*
r_2	`RelPrep`↦*in* `ConstantFn (string mountain.located_in_country)`	Prepositional phrase for multi-hop relation *mountains in the US*
r_3	`RelPrep`↦*in* `ConstantFn (string river.located_in_country)`	Prepositional phrase for multi-hop relation *rivers in the US*
r_4	`RelPrep`↦*in* `ConstantFn (string place.located_in_country)`	Prepositional phrase for multi-hop relation *places in the US*
	Superlative Relations	
r_5	`SuperlativeAdj`↦*longest* `ConstantFn (string river.length)`	Superlative adjectives to describe the length of rivers.
r_6	`SuperlativeMinAdj`↦*shortest* `ConstantFn (string river.length)`	Superlative adjectives to describe the length of rivers.
r_7	`SuperlativeAdj`↦*highest* `ConstantFn (string mountain.elevation)`	Superlative adjectives to describe the elevation of mountains.
r_8	`SuperlativeMinAdj`↦*lowest* `ConstantFn (string mountain.elevation)`	Superlative adjectives to describe the length of rivers.
r_9	`SuperlativeAdj`↦*highest* `ConstantFn (string place.elevation)`	Superlative adjectives to describe the elevation of mountains.
r_{10}	`SuperlativeMinAdj`↦*lowest* `ConstantFn (string place.elevation)`	Superlative adjectives to describe the length of rivers.
r_{11}	`SuperlativeNP`↦*the longest length* `ConstantFn (string river.length)`	Noun phrase used in superlative queries for the length of rivers, *e.g.*, *river in Washington that has the longest length*
r_{12}	`SuperlativeNP`↦*the highest elevation* `ConstantFn (string mountain.elevation)`	For querying the elevations of mountains.
r_{13}	`SuperlativeNP`↦*the highest elevation* `ConstantFn (string place.elevation)`	For querying the elevations of places.

Table 11: List of idiomatic productions used in GEO

C Model Configurations

Paraphraser We finetune the paraphraser using a batch size of $1,024$ tokens for $5,000$ iterations (500 for warm-up), with a learning rate of $3e-5$ using ADAM. We apply label smoothing with a probability of 0.1.

Semantic Parser Our semantic parser is a neural sequence-to-sequence model with dot-product attention (Luong et al., 2015), using a BERT$_{\text{Base}}$ encoder and an LSTM decoder, augmented with copying mechanism. The size of the LSTM hidden state is 256. We decode programs using beam search with a beam size of 5. Following Herzig and Berant (2019), we remove hypotheses from the beam that leads to error executions.

Iterative Training As described in §3.1.1, we first run the iterative paraphrasing and training algorithm for one pass to generate the validation set. In the first iteration of this stage, we train a semantic parser on the (unparaphrased) seed canonical data ($\mathbb{D}_{\text{can}}$) as the initial paraphrase filtering model. In the second stage, we restart the learning process using the generated validation set, and initialize the paraphrase filtering model using the previously trained semantic parser. For each stage, we run the iterative learning algorithm (§2) for two iterations. We generate 10 paraphrases for each example. In each iteration, we train the semantic parser for 30 epochs with a batch size of 64. We use separate learning rates for the BERT encoder ($3e-5$) and other parameters (0.001) in the model (Shaw et al., 2019). For each iteration in the second stage, we perform validation by finding the model checkpoint that achieves the lowest perplexity on the validation set. We perform validation using perplexity for efficiency reasons, as evaluating denotation accuracy requires performing beam search decoding and querying the database, which could be slow.

Evaluation Metric For the perplexity metric to evaluate language gaps, we fine-tune a GPT-2 language model on the paraphrased canonical data $\mathbb{D}_{\text{par}}$ for $1,500$ steps (150 steps for warm-up) with a batch size of 64 and a learning rate of $1e-5$. We use the following equation to compute perplexity

$$\text{PPL}(\mathbb{D}_{\text{nat}}) = \exp\big(\frac{1}{|\mathbb{D}_{\text{nat}}|} \sum_{\langle \boldsymbol{u}, \boldsymbol{z} \rangle \in \mathbb{D}_{\text{nat}}} \frac{-\log p(\boldsymbol{u})}{|\boldsymbol{u}|}\big) \tag{1}$$

This is slightly different from the standard corpus-level perplexity. We use this metric because it is more sensitive (larger Δ) on our small ($< 1K$) evaluation sets, and always correlates with the corpus-level perplexity. For reference, here is the sequence of perplexities using Eq. (1) in the upper half of Tab. 12 compared to the corpus-level ones:

Eq. (1)	22.0	21.4	20.9	20.7	21.5
Corpus-PPL	19.3	18.8	18.4	18.2	18.8

D Does the Model Generalize to Out-of-Distribution Samples?

We also investigated whether the model could generalize to utterances with out-of-distribution program patterns not seen in the training data $\mathbb{D}_{\text{par}}$. We report accuracies on the splits whose program templates are covered (**In Coverage**) and not covered (**Out of Coverage**) by $\mathbb{D}_{\text{par}}$. Not surprisingly, the model performs significantly better on the in-coverage sets with less language mismatch. An exception is $K = 500$ on SCHOLAR, where the perplexity on out-of-coverage samples is slightly lower. This is because utterances in SCHOLAR tend to use compound nouns to specify compositional constraints (*e.g. ACL 2021 parsing papers*), a language style common for in-coverage samples but not captured by the grammar. With smaller K and $\mathbb{D}_{\text{can}}$, it is less likely for the paraphrased data $\mathbb{D}_{\text{par}}$ to capture similar syntactic patterns. Anther factor that makes the out-of-coverage PPL smaller when $K = 500$ is that there are more (simpler) examples in the set compared to $K > 500$, and the relatively simple utterances will also bring down the PPL.

Our results are also in line with recent research in compositional generalization of semantic parsers (Lake and Baroni, 2018; Finegan-Dollak et al., 2018), which suggests that existing models generalize poorly to utterances with novel compositional patterns (*e.g.* conjunctive objects like *Most cited paper by X and Y*) not seen during training. Still surprisingly, our model generalizes reasonably to compositionally novel (out-of-coverage) splits, registering $30\% \sim 50\%$ accuracies, in contrast to HB19 reporting accuracies smaller than 10% on similar benchmarks for OVERNIGHT. We hypothesize that synthesizing compositional samples increases the number of unique program templates in training, which could be helpful for compositional generalization (Akyürek et al., 2021). As an example, the number of unique program templates in $\mathbb{D}_{\text{par}}$ when $K = 2,000$ on SCHOLAR and GEO is $1.9K$ and $1.7K$, resp, compared to only 125 and 187 in $\mathbb{D}_{\text{nat}}$. This finding is reminiscent of data augmentation strategies for supervised parsers using synthetic samples induced from (annotated) parallel data (Jia and Liang, 2016; Wang et al., 2021b).

	K	Acc	In Coverage Acc	In Coverage PPL	Out of Coverage Acc	Out of Coverage PPL
Scholar	500	74.0 ±3.7	82.3	23.4	47.6	18.2
	1,000	75.9 ±1.7	81.4	21.3	50.6	21.7
	2,000	77.8 ±2.2	82.2	20.7	50.2	22.7
	4,000	78.4 ±1.7	83.2	20.5	45.3	22.0
	8,000	75.5 ±2.1	79.8	21.4	43.4	22.4
Geo	500	61.6 ±5.4	79.2	7.6	29.8	9.9
	1,000	68.5 ±7.7	81.4	7.4	28.8	11.3
	2,000	72.8 ±2.8	82.0	7.4	37.6	10.8
	4,000	67.5 ±6.3	75.5	7.6	38.8	11.2
	8,000	67.9 ±4.5	75.5	7.5	41.3	11.2

Table 12: Results on Scholar and Geo with varying amount of canonical examples in the seed training data. We report results on **In Coverage** splits where the program templates of evaluation samples appear in the canonical training data, as well as on **Out of Coverage** splits with disjoint program templates.