SEQZERO: Few-shot Compositional Semantic Parsing with Sequential Prompts and Zero-shot Models

Anonymous ACL submission

Abstract

Recent research showed promising results on combining pretrained language models (LMs) with canonical utterance for few-shot semantic parsing. The canonical utterance is often lengthy and complex due to the compositional structure of formal languages. Learning to generate such canonical utterance requires significant amount of data to reach high performance. Fine-tuning with only few-shot samples, the LMs can easily forget pretrained knowledge, overfit spurious biases, and suffer from compositionally out-of-distribution generalization errors. To tackle these issues, we propose a novel few-shot semantic parsing method – SEQZERO. SEQZERO decomposes the problem into a sequence of sub-problems, which corresponds to the sub-clauses of the formal language. Based on the decomposition, the LMs only need to generate short answers using prompts for predicting sub-clauses. Thus, SEQZERO avoids generating a long canonical utterance at once. Moreover, SEQZERO employs not only a few-shot model but also a zero-shot model to alleviate the overfitting. In particular, SEQZERO brings out the merits from both models via ensemble equipped with our proposed constrained rescaling. SEQZERO achieves SOTA performance on GeoQuery and EcommerceQuery, which are two few-shot datasets with compositional data split.

1 Introduction

Semantic parsing is the transformation of input utterance into formal language, such as SQL query (Zelle and Mooney, 1996), and plays a critical role in NLP applications, such as question answering (Yih et al., 2014), dialogue system (Gupta et al., 2018), and information extraction (Yao and Van Durme, 2014). Training neural semantic parsers requires numerous annotated input utterance and formal language pairs. However, the paired data is usually limited, as the annotation requires experts' knowledge and can be expensive. For example, annotating SQL queries requires programming knowledge, while annotating formal meaning representations like Abstract Meaning Representations (AMR) requires linguistics knowledge. Therefore, semantic parsing in the few-shot setting is a demanding technique.

Researchers have adopted large-scale pretrained language models (LMs, Radford et al. (2019); Brown et al. (2020)) to improve few-shot learning performance. The LMs are usually pretrained on large unlabeled open-domain natural language data and achieves impressive performance on few-shot text-to-text generation problems via proper prompt designing (Brown et al., 2020). Considering the difference between natural and formal language, adapting LMs to semantic parsing is non-trivial. Prior works typically first finetune the LMs to generate canonical utterance, which is then transformed into the final formal language through grammars (Shin et al., 2021; Schucher et al., 2021).

However, the canonical utterance is lengthy and complex due to compositional structure of the formal languages. Learning to precisely generate canonical utterances still requires significant amount of data. Meanwhile, fine-tuning with only few-shot samples, the LMs can easily forget pretrained knowledge, overfit spurious biases, and suffer from compositionally out-of-distribution (OOD) generalization errors. Figure 1 presents an compositionally OOD generalization error of direct fine-tuning BART (Lewis et al., 2019) on the GeoQuery, a dataset about querying in a geographic database. The model incorrectly predicts the table name as "*city*", because the training samples always come from the "*city*" table as long as the query follows the "*how many people live in xxx*" pattern. Such errors account for about 75% of all prediction errors on GeorQuery test set (refer to Section 5.6 for details).

To address the aforementioned issues, we propose a novel prompt-based few-shot learning

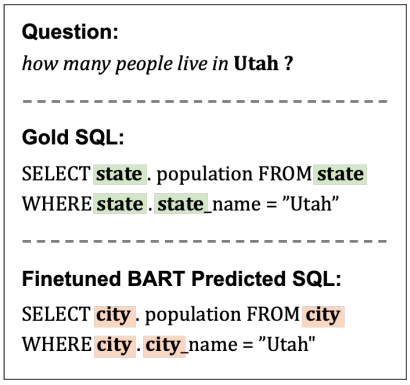

Figure 1: Finetuned BART's OOD generalization errors due to overfitting the spurious biases.

method – SEQZERO. Instead of directly generating the whole formal language, SEQZERO decompose the problem into a sequence of sub-problems, where the LMs only need to make a sequence of short prompt-based predictions. SEQZERO also takes the advantage of zero-shot (un-finetuned) model to avoid overfitting the spurious biases in the training data. Specifically, SEQZERO decompose the problem into predicting the sub-clauses, which make up the formal languages. When predicting a sub-clause, SEQZERO adopts a slot-filling natural language prompt, where the filled prompt can be transformed into the sub-clause through grammars. For filling each prompt, SEQZERO employs two models: a few-shot model and a zero-shot model. Both models ingest the input utterance and the prompt to fill in the slots in the prompt. The few-shot model uses a fine-tuned LM to fill in the slots of each prompt. The zero-shot model directly infers the value in the slots by decoding a pretrained LM with a constrained vocabulary. We then ensemble the prediction from both models, and convert the results for all sub-clauses into the final output (e.g., SQL query). We notice that, the probability mass of the zero-shot model, on the constrained vocabulary, is much smaller than that of the few-shot model. As a result, the zero-shot model cannot take effect in the vanilla ensemble. Therefore, we propose to rescale the probability of the zero-shot model on the constrained vocabulary before ensemble to bring out the advantages of both models.

We conduct experiments on two datasets: GeoQuery, a benchmark dataset that consists of natural language and formal language pairs from geography domain, and EcommerceQuery, a newly collected dataset from E-commerce domain. Results show that our approach outperforms the baseline algorithm and achieves state-of-the-art performance on the compositional split of the two datasets. To sum up, our contributes are:

- We propose to decompose semantic parsing to filling a sequence of prompts, each corresponding to a sub-clause of original SQL query. Compared with direct fine-tuning, predicting sub-clauses is easier, which enables flexible prompt designing and zero-shot model inference.
- We propose the ensemble of few-shot and zero-shot models with help of constrained probability rescaling, which improves out-of-distribution generalization while maintaining in-distribution performance.
- We create and release a new EcommerceQuery dataset [1]. We empirically verify that our approach achieves SOTA on both GeoQuery and EcommerceQuery.

2 Preliminary

Language Modeling aims to estimate the probability distribution for a given sequence of words $x = (w_1, w_2, ..., w_n)$ in an autoregressive way:

$$P_\theta(x) = \prod_{i=1}^{n} P_\theta(w_i | w_1, ..., w_{i-1}),$$

where θ is the parameters of the language model. This approach not only allows estimation of $P_\theta(x)$ but also any conditionals of the form $P_\theta(w_i, w_{i+1}, .., w_n | w_1, ..., w_{i-1})$, which is essentially a seq2seq model. One can leverage a seq2seq model to generate a sequence via a decoding algorithm (e.g., beam-search): $y = \text{Decode}(P_\theta(\cdot|x))$ In recent years, there have been significant progress in training large transformer-based language models (Radford et al., 2019; Brown et al., 2020; Lewis et al., 2019) on large natural language corpus.

Semantic Parsing is to transform an input utterance u into a formal language m. Without loss of generality, we hereafter discuss the case of SQL query as the formal language. One can directly train a language model for semantic parsing:

$$P_\theta(m|u).$$

[1] We will release the code and dataset upon the acceptance of this paper.

Directly learning such a language model is challenging as the difference between the formal language and natural language is huge. Berant and Liang (2014); Shin et al. (2021) propose **S**emantic **P**arsing via **P**araphrasing (SPP), which is a two-stage framework. In the first stage, they paraphrase u to its canonical utterance c using a paraphrasing language model:

$$P_\theta(c|u).$$

In the second stage, the canonical utterance c is transformed into SQL query m by a grammar or a set of rules:

$$m = \text{Grammar}(c).$$

3 Method

In this section, we describe SEQZERO. SEQZERO first decomposes the problem into a sequence of sub-problems as illustrated in Figure 2. For each sub-problem, SEQZERO employs an ensemble of zero-shot and few-shot models to predict a sub-clause of the formal language based on prompts as illustrated in Figure 3.

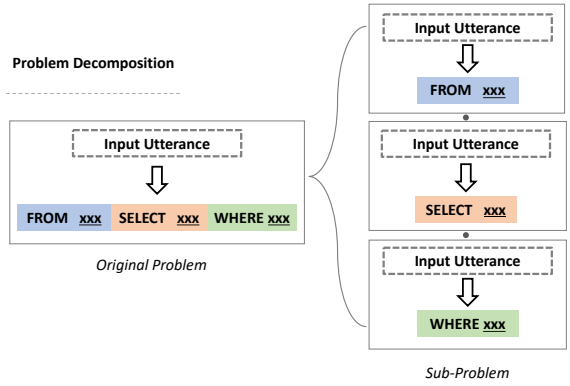

Figure 2: The problem of predicting a SQL can be composed into 3 steps: predicting "FROM" clause, "SELECT" clause, and "WHERE" clause.

3.1 Problem Decomposition and Sequential Prompt Filling

Each SQL query can be regarded as a composition of different types of sub-clauses, such as "SELECT", "FROM", "WHERE":

$$m = \text{Compose}(m_1, ..., m_n),$$

where m_i is the sub-clause of the i-th type, n is the number of all possible types of sub-clauses, and the composition is conducted via a rule-based system. A simple example of the composition function is direct concatenating the sub-clauses, whereas the real implementation requires some dedicated design. For example, m_i can be a null clause, e.g., not every SQL query contains a "WHERE" clause. We discuss the implementation details of the composition in Appendix C.

We turn the problem of direct predicting m into predicting m_i sequentially from m_1 to m_n. We remark that the prediction of m_i depends on $m_1, ..., m_{i-1}$, as illustrated in Figure 3. Similar to the SPP framework, we design a canonical utterance c_i for each sub-clause m_i. The transformation between c_i and m_i is conducted by a grammar:

$$m_i = \text{Grammar}(c_i).$$

Each c_i consists of two parts: a natural language slot-filling prompt p_i and a value in the slot v_i:

$$c_i = \text{FillSlot}(p_i, v_i).$$

The prompt p_i is shared across all sub-clauses of the i-th type, while the value v_i varies for different instances. As a result, the problem is turned into predicting the values $\{v_i\}_{i=1}^n$ given the input utterance u, and prompts $\{p_i\}_{i=1}^n$ sequentially from $i = 1$ to $i = n$. The prediction is conducted via decoding a language model, $P_{\theta_i}(\cdot|u, m_1, \ldots, m_{i-1}, p_i)$, where the canonical utterances of previous sub-clauses $(m_1, \ldots, m_{i-1})$ are also provided as the extra context. We summarize the process in Algorithm 1.

Algorithm 1: Sequential Prompt Filling

Input: u: input utterance; $\{p_i\}_{i=1}^n$: prompts; Grammar: grammar for parsing the canonical utterance; $\{P_{\theta_i}\}_{i=1}^n$: LMs.

for $i = 1, \cdots, n$ **do**
- $x = (u, m_1, \ldots, m_{i-1}, p_i)$
- $v_i = \text{Decode}(P_{\theta_i}(\cdot|x))$
- $c_i = \text{FillSlot}(p_i, v_i)$
- $m_i = \text{Grammar}(c_i)$

end

$m = \text{Compose}(m_1, ..., m_n)$

Output: m: SQL query

3.2 Ensemble of Few-shot and Zero-shot Models

Despite the apparent advantages of sequential prompt filling, directly fine-tuning LMs on few-

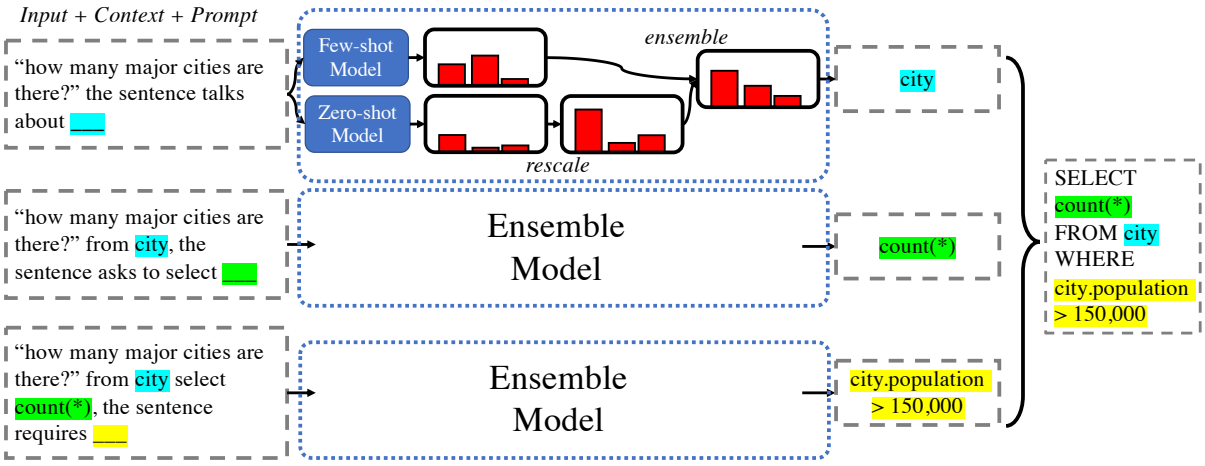

Figure 3: Pipeline of sequential prompt filling and SQL generation on GeoQuery. Note that, the scale of the prediction probability of the zero-shot model is very small before rescaling.

shot samples will fall short due to the overfitting. Because of the better OOD generalizability of zero-shot models, we propose to employ the ensemble of a few-shot model $P_{\theta_{i,f}}$ and a zero-shot model $P_{\theta_{i,z}}$ for each language model P_{θ_i}.

Few-shot Model. Each few-shot model is obtained by finetuning a pretrained language model via minimizing the negative log-likelihood loss:

$$\arg\min_{\theta_{i,f}} -\log P_{\theta_{i,f}}(v_i|u, m_1, \ldots, m_{i-1}, p_i),$$

where $v_i, m_1, \ldots, m_{i-1}$ are the ground truth from the few-shot training data. It is essentially the teacher forcing training strategy. Note that we omit the summation over the training set for simplicity and clarity.

Zero-shot Model. Each zero-shot model directly adopts the pretrained language model P_{θ_0}. Without any guidance, P_{θ_0} may generate any free text even if we provide the input utterance and prompt. In order to mine the knowledge from P_{θ_0}, we only allow the zero-shot model to generate from a list of candidate values. The candidate values are collected from multiple sources including SQL grammar, table schema, input utterance and training data. When predicting the j-th word for v_i, the zero-shot model rescales the probability on a constraint vocabulary, which is specifically designed for the i-th clause:

$$P_{\theta_{i,z}}(w|x) = \frac{\mathbb{1}(w \in V_i(x))P_{\theta_0}(w|x)}{\sum_{w_j \in V_i(x)} P_{\theta_0}(w_j|x)}, \quad (1)$$

where w is a predicting word, $x = (u, m_1, ..., m_{i-1}, p_i, w_1, .., w_{j-1})$ is the context for predicting the i-th value, $\{w_t\}_{t=1}^{j-1}$ is the prefix in the value, $V_i(x)$ is the constraint vocabulary. Given the list of candidate values, we use a trie (prefix tree) to compute all the allowed tokens, and thus $V_i(x) = V_i(\{w_t\}_{t=1}^{j-1})$ depends on the prefix of the values. Note that, to develop a more flexible method, a trie/prompt could start at intermediate steps.

Ensemble. We then obtain P_{θ_i} by a linear ensemble of the few-shot model $P_{\theta_{i,f}}$ and the zero-shot model $P_{\theta_{i,z}}$:

$$P_{\theta_i} = \gamma_i P_{\theta_{i,f}} + (1 - \gamma_i) P_{\theta_{i,z}}, \quad (2)$$

where γ_i is a clause-specific weight for trade-off between two models.

Remark. We employ a normalization step in the zero-shot model Eq. (1). The normalization is not necessary for the zero-shot model itself, but plays a critical role in the ensemble. This is because the scales of the predicted probabilities of few-shot and zero-shot models are different, as illustrated in Figure 3. The P_{θ_0}'s prediction probability is distributed over the whole vocabulary. There is only a very small probability mass assigned to the allowed tokens, $V_i(x)$. On the other hand, the few-shot model's prediction probability is almost entirely distributed over $V_i(x)$. Without rescaling, the zero-shot model will only have little effect when ensembling with the finetuned model.

4 Experiment Setup

Dataset To evaluate the performance of our proposed method, we conduct experiments on the Geo-

Query dataset (Zelle and Mooney, 1996), where there are 880 queries to a database of U.S. geography. To test compositional generalizability, we adopted the compositional split for SQL released by Finegan-Dollak et al. (2018), where templates created by anonymizing entities are used to split the original dataset, to make sure that all examples sharing a template are assigned to the same set. There are 536/159/182 examples for train/dev/test set, thus this setting can be regarded as the few-shot setting. We also experimented with even fewer training examples (50, 150).

Besides, we create and release the EcommerceQuery, a new SQL semantic parsing dataset in E-commerce domain. Specifically, we collect natural language utterances from user input search queries to an e-commerce website. To create corresponding SQL queries, we use some self-defined rule with manual audition. We construct compositional splits, where there are unseen SQL query patterns in the dev/test set. Finally, train/dev/test set contains 1,050/353/355 examples respectively. For details, please refer to Appendix B. Two examples from EcommerceQuery are shown in Table 7.

Baselines and Models We use seq2seq finetuned BART as our main baseline on both datasets. Without explicit notations, we use BART large in all of the following experiments. On GeoQuery dataset, we use prior state-of-the-art methods as additional baselines. On EcommerceQuery dataset, we use only LSTM seq2seq and BART as baselines, because Iyer et al. (2017) requires user feedbacks, and Zheng and Lapata (2020) requires domain specific semantic tags, which are not available in EcommerceQuery.

Evaluation Following Andreas (2019), we use exact-match accuracy as the evaluation metric, namely the percentage of examples that are correctly parsed to their SQL queries.

5 Experimental Results

5.1 Main Results

Table 1 shows our main results on GeoQuery and EcommerceQuery datasets. As shown in Table 1, on GeoQuery dataset, the finetuned BART beats all the previous baseline methods. Our approach outperforms all baseline systems by a substantial margin, reaching new SOTA performance. Note that directly combining BART with the semantic parsing via paraphrasing (SSP) framework even decrease the performance of BART, because paraphrased canonical utterances for SQL on GeoQuery is too long and complex to directly generate. Even comparing with Zheng and Lapata (2020), SEQZERO achieves a much better performance without the usage of anonymized variables [2]. In addition, on EcommerceQuery dataset, our SEQZERO further achieves considerable improvements over the baseline methods, reaching SOTA performance. Comparing with BART, the best baseline model, SEQZERO gains improvement in exact-match accuracy by 8.5%. In all words, our model is an extremely strong performer and substantially outperforms baseline methods, which demonstrate the efficiency of our method.

Method	GeoQuery	EcoQuery
Iyer et al. (2017) †	40.0	-
Andreas (2019) †	49.0	-
Zheng and Lapata (2020) †◇	69.6	-
Our Implementation		
LSTM seq2seq	39.0	9.3
BART	72.5	37.7
BART + SPP	66.5	37.2
SEQZERO	**74.7**	**46.2**

Table 1: Results on GeoQuery test set of compositional split, and on EcommerceQuery (EcoQuery) dataset. †: we directly report the metrics in the original papers, while our reproduction achieves similar performance. ◇: Zheng and Lapata (2020) took an unfair advantage of anonymized variables.

5.2 Ablation Study

To demonstrate the utility of sequential prompt filling and zero-shot model, we conduct a set of ablation experiments, as shown in Table 2. In each ablation experiment, we delete one of these two key components of SEQZERO, namely "−SEQ" and "−ZERO".

SEQZERO −ZERO means that we directly use finetuned few-shot models to fill in sequential prompts without using the zero-shot model.

SEQZERO −SEQ is equivalent to the ensemble of a finetuned BART and a un-finetuned BART for

[2] Zheng and Lapata (2020) could not directly compare with our method, because they use anonymized variables (i.e. oracle entities), while other models including SEQZERO require generating entities instead of using oracle entities. Thus, for fair comparison, their method without variable anonymization would have even worse performance, indicating even larger improvements of our method.

Method	GeoQuery	EcoQuery
SEQZERO	**74.7**	**46.2**
−SEQ	74.2	44.5
−ZERO	71.4	37.7

Table 2: Ablation study of SEQZERO.

predicting the SQL query directly without sequential prompt filling.

On both datasets, "−SEQ" decreases the performance of SEQZERO. It indicates that designing clause-specific prompt can better mine the pretrained knowledge from the language model. Meanwhile, zero-shot model ensemble brings our model better out-of-distribution generalization ability. Consequently, when zero-shot model ensemble is ablated, the performance drops a lot ("−ZERO" vs "SEQZERO").

5.3 Analysis on Different Clauses

Here, we try to understand how the model performs on different clauses. We report the prediction accuracies of SEQZERO and "−ZERO" on 5 clauses on the GeoQuery dataset in Table 3. We can clearly see that SEQZERO has better performance by leveraging the zero-shot model on all clauses.

Method	FROM	SELECT	WHERE	GROUP	ORDER
SEQZERO	**88.5**	**77.5**	**74.7**	**74.7**	**74.7**
−Zero	84.1	74.2	71.4	71.4	71.4

Table 3: Prediction accuracies on all 5 clauses on GeoQuery dataset.

Recall that the prediction of the latter clauses depends on the previous ones, the performance of each next clause generally decreases due to error propagation. The same performance of "WHERE", "GROUP" and "ORDER" is because there are very few "GROUP" and "ORDER" clauses on test set. As can been seen, SEQZERO achieves much better performance on the "FROM" clause and thus significantly reduces the error propagation.

5.4 Impact of Prompt Designing

Table 4 shows the performance of the few-shot finetuned BART and the zero-shot BART (in constrained decoding setting) with several representative prompts on "FROM" clause of GeoQuery test set. We can see that prompt designing highly affects the the zero-shot model's performance, while

Prompt	Few	ZERO
the answer can be obtained from	81.3	65.9
the sentence talks about	**84.1**	**78.0**

Table 4: Impact of prompt designing for few-shot Few and zero-shot ZERO BART on "FROM" clause of GeoQuery test set.

Prompt	attribute+relation	relation
the sentence requires	39.2	49.3
where	21.1	51.5
the condition is :	**51.1**	**57.3**

Table 5: Impact of prompt designing for zero-shot BART on "CONDITION" clause of EcommerceQuery test set. In attribute+relation setting, we let zero-shot model generate both attributes and relations. In relation setting, we let zero-shot model generate relations only.

it has less impact on few-shot finetuned model. Table 5 shows the performance of the zero-shot BART on "CONDITION" part of EcommerceQuery test set, where different prompts also lead to significantly different performance. These results reveal the necessity of sequential prompt filling. Without this component, one cannot easily come up with a proper prompt for achieving a better model performance. In practice, we design 20 prompt sets and select the best one based on the zero-shot model's performance on the developing set.

5.5 Impact of Training Data Size

Table 6 shows the performance of baseline BART and our SEQZERO (as well as ablation of ZERO), facing different numbers of training data points in the few-shot setting. With 50, 150 training samples, we make sure that each SQL query template occurs only once to maximize the diversity of training data. For the full dataset, there are 536 samples with 158 different training templates in total.

Our SEQZERO outperforms BART in all settings (50, 150, 536 training samples), which shows the

# of Samples	50	150	536
BART	41.2	73.1	72.5
SEQZERO	**48.9**	**74.2**	**74.7**
−ZERO	31.3	73.1	71.4

Table 6: Model accuracy with different numbers of training samples on GeoQuery dataset.

effectiveness of our method in the few-shot setting. From 50 to 150 training samples, the model see more SQL templates, which help compositional generalization, and lead to the increased performance of all models. From 150 to 536 samples, the performance of BART and "−ZERO" decrease slightly. That is because there are multiple samples of the same templates in the 536 training samples, and the models overfit to those training templates. In contrast, SEQZERO avoids such overfitting with the help of zero-shot models and achieves better performance by leveraging more training samples.

Without the aid of zero-shot model, "−ZERO" performs worse than SEQZERO. When there are only 50 samples, the performance degradation is the most significant. When there are 536 samples, the decrease led by ablation of zero-shot model is larger than that of 150 samples. It is because when there are many cases for each template, ensemble of zero-shot model can alleviate overfitting such templates.

Furthermore, "−ZERO" has similar performance with BART when there are over 150 training samples. On the other hand, the performance of "−ZERO" is worse than BART when there are very few training samples (50 samples). We conjecture that this is because BART shares the model parameter between all sub-clauses, while "−ZERO" finetunes models separately on different sub-clauses. The parameter sharing will further lead to knowledge sharing across sub-clauses and improves the performance. How to leverage the benefit from both parameter sharing and SEQZERO could be an interesting future research topic.

5.6 Case Study

Table 7 shows BART and SEQZERO's predictions for some cases. For first example, BART gives a wrong prediction, because few-shot training samples introduce too many spurious biases to the finetuned model. In contrast, SEQZERO gives correct prediction. Actually, after analyzing the errors made by finetuned BART models on GeoQuery, among all errors on test set, the common error for around 75% examples is the table name error in "FROM" clause, which is due to spurious biases.

For the second example, BART predicts "PRICE <" incorrectly even seeing "*over*", because EcommerceQuery Dataset is designed to include only "PRICE <" but no "PRICE >" template. Our SEQZERO could give the correct prediction because of better OOD generalizability with the help of zero-shot models.

Even with our SEQZERO, there are still many errors. For instance, in the third example, it still struggles with identifying the size in the natural language query and generating the Size filtering condition in WHERE clause.

6 Related Work

Few/Zero-shot Semantic Parsing Shin et al. (2021); Schucher et al. (2021) conducted few-shot semantic parsing by using pretrained LMs to first generate canonical natural language utterances, and then transform them to final formal language through synchronous context-free grammar (SCFG) (Jia and Liang, 2016). However, dealing with complex structure and lengthy canonical language is still challenging for models in the few-shot setting. Also, canonical languages created through SCFG allows limited space for prompt designing, and canonical language's form is still too strange for language models to understand. Zhong et al. (2020) explored zero-shot semantic parsing via generation-model-based data augmentation. Other ways of bootstrapping a semantic parsing requires rules/grammars to synthesize training examples (Xu et al., 2020; Wang et al., 2015; Yu et al., 2020; Campagna et al., 2019; Weir et al., 2020; Marzoev et al., 2020; Campagna et al., 2020).

Semantic Parsing via Paraphrasing Berant and Liang (2014) started the line of work where semantic parsing is finished through an intermediate paraphrasing step. Wang et al. (2015); Marzoev et al. (2020) generated paraphrase candidate values from a grammar of legal canonical utterances, and incrementally filtered the bottom-up or top-down generation by scoring the partial candidates against final formal language. All such work did not exploit the power of pretrained models to generate intermediate paraphrases.

Compositional Generalization in Semantic Parsing Compositional generalization is an essential problem in semantic parsing because formal languages are internally compositional. Generally, one way to improve compositional generalizability is to incorporate inductive biases directly to models through moduler models (Dong and Lapata, 2018), symbolic-neural machines (Chen et al., 2020), latent variables/intermediate representations (Zheng and Lapata, 2020; Herzig and Berant, 2020), meta-

Cases	Text
Question	*what is the population of utah*
BART	SELECT city . population FROM city WHERE city . city_name = "utah"
SEQZERO	SELECT state . population FROM state WHERE state . state_name = "utah"
Ground Truth	SELECT state . population FROM state WHERE state . state_name = "utah"
Question	*petrol trimmer over 100 dollar*
BART	SELECT * FROM ASINs WHERE Maching Algorithm("petrol trimmer") == True and Price < 100
SEQZERO	SELECT * FROM ASINs WHERE Maching Algorithm("petrol trimmer") == True and Price > 100
Ground Truth	SELECT * FROM ASINs WHERE Maching Algorithm("petrol trimmer") == True and Price > 100
Question	*mi4 64 gb mobile phone*
BART	SELECT * FROM ASINs WHERE Maching Algorithm("mi4 64 gb mobile phone") ORDER BY date
SEQZERO	SELECT * FROM ASINs WHERE Maching Algorithm("mi4 64 gb mobile phone") ORDER BY date
Ground Truth	SELECT * FROM ASINs WHERE Maching Algorithm("mi4 mobile phone") and Size = 64 gb

Table 7: Cases studys. The first example is from GeorQuery, and the last two examples are from EcoQuery.

learning (Lake, 2019) etc. Another way is to first do data augmentation and then train a model with augmented data (Andreas, 2019; Zhong et al., 2020; Yu et al., 2020; Akyürek et al., 2020). Pretrained models has also been shown useful for compositional semantic parsing (Oren et al., 2020; Furrer et al., 2020). None of prior work used decomposition and prompt filling, or zero-shot models to improve compositional generalizability.

Prompting for Few/Zero-shot learning Natural language prompts are widely used in few-shot or zero-shot learning. There are several fashions to use prompts in Autoregressive Language Models (Liu et al., 2021a). One is tuning-free prompting, for example, Petroni et al. (2019); Shin et al. (2020) used a fill-in-the-blank paradigm, while Brown et al. (2020); Shin et al. (2021) used "few-shot" prompts that included several examples of inputs followed by target outputs, with the actual task input appended at the end. One is Fixed-LM Prompt Tuning, as used by Li and Liang (2021); Schucher et al. (2021); Qin and Eisner (2021); Liu et al. (2021b), which requires training less parameters compared with tuning the whole model. Another is Fixed-prompt LM Tuning, which is similar to our setting. We choose to use this way because it is demonstrated better than other methods in many few-shot NLP tasks (Gao et al., 2020) when tuning the whole model is not a concern. This is also more efficient at inference time, as it is no longer necessary to select training examples to precede the test input. Note that, (Mishra et al., 2021) employed prompt decomposition during tuning-free prompting, which is validated in other NLP tasks.

Zero-shot pretrained models for OOD generalization Wortsman et al. (2021) showed that, in computer vision tasks, although fine-tuning approaches substantially improve accuracy in-distribution, they reduce out-of-distribution robustness, while zero-shot pretrained models have higher OOD generalizability. Thus, model weight ensemble (Wortsman et al., 2021) and model editing (Mitchell et al., 2021) were leveraged to manipulate zero shot pretrained models, which motivets us to ensemble zero-shot and few-shot models during the generation process of semantic parsing. We tried weight ensembling proposed by Wortsman et al. (2021), but it does not work in our generation setting. The reason is the same as why directly ensembling in prediction space is not working. That's said, weights in a zero-shot model correspond to the probability over the whole vocabulary while weights in a finetuned model correspond to the probability over constrained vocabulary. Thus, weights in the zero-shot model have little effect on the constrained vocabulary.

7 Conclusion

Although prior work leveraged pretrained LMs and canonical language for few-shot semantic parsing, generating lengthy and complex canonical language is still challenging, leading finetuned models to overfitting spurious biases in few-shot training examples and demonstraining poor compositional generalizability. To tackle this, we propose to filling in sequential prompts with LMs and then compose them to obtain final SQL queries. During the process, our proposed zero-shot pretrained model ensembling or uncertainty-based model selection could significantly boost the performance on critical clauses, leading to overall SOTA performance on GeoQuery and our released EcommerceQuery semantic parsing dataset.

Ethical Impact

SEQZERO is a general framework for few-shot semantic parsing on text, such as search queries. SEQZERO neither introduces any social/ethical bias to the model nor amplify any bias in the data. When creating EcommerceQuery dataset, we collected data on an E-commerce search platform without knowing customers' identity. No customer/seller specific-data is disclosed. We build our algorithms using public code bases (PyTorch and FairSeq). We do not foresee any direct social consequences or ethical issues.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2020. Learning to recombine and resample data for compositional generalization. *arXiv preprint arXiv:2010.03706*.

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

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica S Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. *arXiv preprint arXiv:2005.00891*.

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.

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

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

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

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. *arXiv preprint arXiv:1810.07942*.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. *arXiv preprint arXiv:1906.05381*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Alana Marzoev, Samuel Madden, M Frans Kaashoek, Michael Cafarella, and Jacob Andreas. 2020. Unnatural language processing: Bridging the gap between synthetic and natural language data. *arXiv preprint arXiv:2004.13645*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2021. Reframing instructional prompts to gptk's language. *arXiv preprint arXiv:2109.07830*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

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

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nathan Schucher, Siva Reddy, and Harm de Vries. 2021. The power of prompt tuning for low-resource semantic parsing. *arXiv preprint arXiv:2110.08525*.

Richard Shin, Christopher H Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. *arXiv preprint arXiv:2104.08768*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

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.

Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, et al. 2020. Dbpal: A fully pluggable nl2sql training pipeline. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2347–2361.

Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. 2021. Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903*.

Silei Xu, Sina J Semnani, Giovanni Campagna, and Monica S Lam. 2020. Autoqa: From databases to qa semantic parsers with only synthetic training data. *arXiv preprint arXiv:2010.04806*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648.

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.

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. *arXiv preprint arXiv:2009.07396*.

A Configuration

A.1 Training Details

During training, we use fairseq (Ott et al., 2019) to implement BART model. We use Adam as optimizer with a learning rate 1e-5. We use dropout and attention dropout with 0.1 as dropout rate. Also, we use label smoothing with a rate 0.1. Batch sizes are 1024 tokens. Besides, we employ a weight-decay rate 0.01. All the parameters are manually tuned based on the dev performance.

We train all models on NVIDIA A100 SXM4 40 GB GPU. We set the max training epoch to be 100 and select the best performed epoch according to dev performance. Training process on each clause or whole sequence could be finished within 3 hours.

A.2 Inference Details

During inference, we use greedy search to decode. We also use ensemble of zero-shot and few-shot models during this process. The ensemble weight γ_i in Eq. (2) is chosen from [0, 1] and tuned by grid search according to performance on dev set.

B EcommerceQuery Dataset

When we create the EcommerceQuery dataset, we first we collect natural language utterances from user input search queries to an e-commerce website. To create corresponding SQL queries, we use regular expressions to create "SIZE" filtering conditions, and use some rules to create "PRICE" filtering conditions, "DELIVERY" attributes and "SUBSCRIBE" attributes in "WHERE" clauses. Finally, we manually audit each pair of data to ensure the quality.

To construct compositional splits, we make sure that there is no "PRICE>", "SIZE=", and "SUBSCRIBE=" SQL templates in training set but the majority of SQL queries on dev and test set contains such templates. Ideally, a model with good compositional generalizability could generalize from "PRICE<" and "SIZE>" to "PRICE>", generalize from "PRICE=" and "SIZE>" to "SIZE=", and generalize from "DELIVERY=" to "SUBSCRIBE=".

C Problem Decomposition on GeoQuery and EcommerceQuery

In this section we introduce the problem decomposition for GeoQuery and EcommerceQuery in details. We answer the following two questions: 1. what are the sub-clauses in the sub-problems? 2. how to compose the final formal language from the sub-clauses.

C.1 GeoQuery

On GeoQuery, there are totally 5 sub-clauses, namely FROM, SELECT, WHERE, GROUP-BY, ORDER-BY clauses. we first generate FROM from clause with the prompt "*the sentence talks about*". Then we generate SELECT clause with the prompt "*the sentence talks about*", generate "*Where* clause with the prompt THE SENTENCE REQUIRES", generate GROUP-BY clause with the prompt THE SENTENCE REQUIRES TO GROUP BY, and generate ORDER-BY clause with the prompt "*the sentence requires the result to be ordered by*" Note that prior generated clauses are used as additional prefix to generate current clauses. The filled value for each clause could be "*None*". When the filled value is "*None*", which means there is no such clause in the final SQL query. Finally, we compose all clauses (if the filled value is not "*None*") sequentially to obtain the final SQL query.

C.2 EcommerceQuery

On EcommerceQuery, there are totally 2 sub-clauses, namely MATCHING, and CONDITION clauses. Because thes two clauses are less dependent, we generate each clause separately and then compose the generated values of each clause. When generating MATCHING clause, we use the prompt "*matching algorithm (*". When generating CONDITION clause, we use the prompt "the condition is :".