Improving Generalization in Semantic Parsing by Increasing Natural Language Variation

Anonymous ACL submission

Abstract

The development of Spider (Yu et al., 2018), a large-scale dataset with complex programs and databases from several domains, has led to much progress in text-to-SQL semantic parsing. However, recent work has shown that models trained on Spider often struggle to generalize, even when faced with small perturbations of previously seen expressions. This is mainly due to the linguistic form of questions in Spider which are overly specific, unnatural, and display limited variation. In this work, we use data augmentation to enhance the robustness of text-to-SQL parsers against natural language variations. Existing approaches generate question reformulations either via models trained on Spider or only introduce local changes. In contrast, we leverage the capabilities of large language models to generate more realistic and diverse questions. Using only a few prompts, we achieve a two-fold increase in the number of questions in Spider. Training on this augmented dataset yields substantial improvements on a range of evaluation sets, including robustness benchmarks and out-of-domain data.[1]

1 Introduction

Semantic parsing is the task of mapping natural language utterances to machine-interpretable expressions such as SQL queries or logical forms. It has emerged as an important component in many natural language interfaces (Őzcan et al., 2020) with applications in robotics (Dukes, 2014), question answering (Zhong et al., 2017; Yu et al., 2018), dialogue systems (Artzi and Zettlemoyer, 2011), and the Internet of Things (Campagna et al., 2017).

The release of the Spider dataset (Yu et al., 2018) marked an important milestone in text-to-SQL semantic parsing. Apart from its considerable size, Spider stands out for including complex and nested queries, and databases from various domains. Importantly, it exemplifies a cross-domain generalization setting, i.e., models trained on Spider are expected to parse natural language questions for any given database, even in previously unseen domains. In practice, models trained on Spider degrade significantly when tested on different databases from *other* datasets, for example, on real-world data from Kaggle and Stack Exchange websites (Suhr et al., 2020; Lee et al., 2021; Hazoom et al., 2021).

The linguistic composition of questions in Spider contributes to this performance gap. Unlike real-world applications where user questions may be concise, ambiguous, and necessitate commonsense reasoning or domain-specific knowledge, questions in Spider are often overly explicit, directly mentioning database entities even when such information is unnecessary for inferring the underlying intent. An example is shown in Figure 1, the first question includes redundant details (e.g., *customer*, *first name*, *last name*) which serve as references to databases entities. Omitting these details would not change the meaning of the question but rather make it more colloquial. Due to the limited diversity of questions, Spider falls short in providing enough examples for learning essential skills such as grounding and reasoning. As a result, models tend to overfit to Spider-style questions, and even minor perturbations in how questions are phrased lead to a considerable performance decrease, sometimes up to 22% (Gan et al., 2021b; Deng et al., 2021; Pi et al., 2022; Chang et al., 2023).

More realistic training sets can potentially alleviate generalization problems but are challenging to create because semantic parsing requires annotators familiar with the specific meaning representation language being used (e.g., SQL). At the time of writing, Spider (Yu et al., 2018) remains the largest and most extensively used dataset for text-to-SQL tasks. Efforts to automatically increase its diversity often rely on text generation models trained on the same Spider data and unavoidably inherit its char-

[1] Model checkpoints and data are available at URL.

acteristics (Zhong et al., 2020; Wang et al., 2021; Wu et al., 2021; Jiang et al., 2022).

In this work, we propose to augment the training data for text-to-SQL parsers with more realistic and diverse question reformulations. We leverage the capabilities of large language models for rewriting utterances and devise prompts designed to enhance model robustness against linguistic variations. We train three state-of-the-art parsers on Spider (Yu et al., 2018) with augmentations generated by our approach. Extensive experiments show that a two-fold increase in the number of questions substantially improves model generalization ability. Our augmentations increase *robustness* against question perturbations when models are evaluated on the challenging Dr.Spider sets (Chang et al., 2023) and deliver improvements in a *zero-shot* setting, when models are tested on out-of-domain datasets like GeoQuery (Zelle and Mooney, 1996) and KaggleDBQA (Lee et al., 2021).

Our contributions are three-fold: a proposal of rewrite operations to render questions more diverse and natural; a methodology for augmenting existing datasets based on the proposed reformulations; and empirical results validating our approach improves generalization across models and datasets.

2 Related Work

Out-of-domain Generalization Several datasets have been released to facilitate the development of models with generalization capabilities. WikiSQL (Zhong et al., 2017) is a large-scale benchmark with different databases but only one table. As a result, WikiSQL queries are relatively easy to parse due to the use of a limited set of operations. Spider (Yu et al., 2018), contains multiple tables per database which result in complex SQL queries.

Suhr et al. (2020) examine the performance of Spider-trained models on datasets varying in terms of the questions being asked, the database structure, and SQL style. They discover that a key challenge in achieving generalization lies in linguistic variation, and propose augmenting Spider's training set with WikiSQL data. Our work addresses the problem of question diversity in Spider, without compromising its complex query structures or multi-table database nature. We evaluate our approach on GeoQuery (Zelle and Mooney, 1996), a dataset similar to Spider in terms of database structure and SQL queries but different in the style of questions. We also report results on KaggleDBQA (Lee et al., 2021), a dataset with real-world databases and questions created by users with access to field descriptions rather than database schemas.

Robustness to Perturbations Another challenge for text-to-SQL parsers is robustness to small perturbations. Previous studies evaluate robustness in the single-domain setting (Huang et al., 2021) and across databases, e.g., by removing or paraphrasing explicit mentions of database entities (Spider-Realistic; Deng et al. 2021) or by substituting such mentions with synonyms (Spider-Syn; Gan et al. 2021a). Other work explores the effect of perturbations in the database schema (Pi et al., 2022) and also in questions (Ma and Wang, 2021). Recently, Chang et al. (2023) released Dr.Spider, a comprehensive robustness benchmark with a wide range of perturbations in the database schema, questions, and SQL semantics. We evaluate our approach on their "question sets" which cover a broader range of language variations compared to previous efforts.

Data Augmentation Several data augmentation and adversarial training techniques have been proposed to support SQL queries executed on a single table (Li et al., 2019; Radhakrishnan et al., 2020) and multiple tables (Zhong et al., 2020; Wang et al., 2021; Wu et al., 2021; Deng et al., 2021; Wu et al., 2021; Jiang et al., 2022). Augmentations in earlier work (Gan et al., 2021a; Deng et al., 2021; Ma and Wang, 2021; Huang et al., 2021) target specific linguistic expressions like synonyms or paraphrases. We leverage the capabilities of (very) large languages models (LLMs; Brown et al. 2020; Chowdhery et al. 2022) to generate linguistically diverse natural language questions. Recent efforts (Dai et al., 2023; He et al., 2023) have shown that LLMs can serve as annotators when given sufficient guidance and examples mainly for text classification, while we focus on semantic parsing.

3 Motivation

3.1 Problem Formulation

Semantic parsing aims to translate a natural language utterance into a formal representation of its meaning. We focus on meaning representations in the form of SQL queries that can be executed in some database to retrieve an answer or denotation. In the cross-domain setting, the parser is not limited to a specific database and can be in theory applied to arbitrary databases and questions. In practice, this task is more or less complex depending on the database in hand, i.e., the number of tables and values, the naming conventions used for tables and columns, the way values are formatted, and specific domain characteristics. We do not consider these challenges in this work, focusing instead on generalization issues that arise from the variation of questions in natural language.

3.2 Types of Utterances in Semantic Parsing

Recent work has demonstrated the importance of wording in semantic parsing, indicating that certain question formulations can be more difficult to parse than others (Radhakrishnan et al., 2020; Gan et al., 2021a; Deng et al., 2021; Chang et al., 2023).

The level of difficulty for a question can be influenced by the amount of task-specific background knowledge used to formulate it. For instance, users familiar with SQL and the underlying database will have some idea of the desired program, and will be able to articulate their intentions more precisely, e.g., by providing explicit instructions. In contrast, users unfamiliar with the task are more likely to ask general questions in a colloquial style. Figure 1 illustrates different question formulations with the same intent. The first question could have been posed by a user who is well-versed in SQL and has knowledge of the database; it mentions specific database entities and operations like summation and filtering, unlike the second question which does not have any such details. More formally, we distinguish between two types of utterances:

Utterances which demonstrate prior knowledge are closely aligned with the desired programs, highlight logical structure operations, and explicit references to database entities. Such utterances resemble instructions, suggesting the user has some understanding of the desired program. In Figure 1, the first question falls under this category, presupposing knowledge of summation and filtering operations and the names of entities (e.g., `first_name`, `last_name`) used in the target SQL query.

Utterances which do not demonstrate prior knowledge are general descriptions of intent, expressed in a simple, colloquial language. They do not provide intentional hints about the desired program, but are often ambiguous, requiring additional reasoning based on domain or common sense knowledge. In the examples shown in Figure 1, the second question belongs to this category, it is laconic, underspecified, and inherently natural.

Database: driving_school

Customers					
customer_id	...	first_name	last_name	...	email_address

Lessons					
lesson_id	...	customer_id	lesson_time	...	price

Questions	Prior SQL	Prior DB
1. Calculate the total sum of lesson times filtering the results by selecting the customer with the first name "Rylan" and the last name "Goodwin".	✓	✓
2. How long did Rylan Goodwin's lesson last?	X	X
3. How long is the total lesson time taken by a customer with a first name as Rylan and a last name as Goodwin?	X	✓

SQL Query

```
SELECT sum(T1.lesson_time) FROM Lessons AS T1 JOIN
Customers AS T2 ON T1.customer_id = T2.customer_id WHERE
T2.first_name = "Rylan" AND T2.last_name = "Goodwin".
```

Figure 1: Different types of questions that are related to the same database (only relevant tables and columns are shown) and map to the same SQL query.

These types of utterances represent two important edge cases but do not cover all possibilities. In the context of text-to-SQL semantic parsing, information about the database schema and its contents can also be useful when formulating questions. We thus introduce a third category that falls between having task-specific knowledge and none at all.

Utterances which demonstrate knowledge of the database schema are general descriptions of intent but with explicit references to related database entities. This category differs from the previous two in the type of prior knowledge used; users are familiar with the database schema and possibly database content but have no expertise in query construction. The third question in Figure 1 includes explicit references to the database table (e.g., `customers`) and its columns (e.g., `lesson_time`, `first_name`, `last_name`). Because of that, questions may be less coherent and natural. In our example, the question contains redundant details such as *first name*, *last name*, and *customer*.

Questions in Spider (Yu et al., 2018) often include explicit mentions of database elements (Deng et al., 2021). This is a by-product of Spider's creation process which encouraged annotators familiar with SQL to formulate the questions more clearly and explicitly. In contrast, other datasets like GeoQuery (Zelle and Mooney, 1996) or cross-domain KaggleDBQA (Lee et al., 2021) contain less explicit questions with a smaller percentage of database entity mentions. In this work, we automatically augment Spider's training set with more general and natural questions aiming to develop semantic parsing models that can effectively handle all types of utterances mentioned above.

4 Data Generation

We augment the training set of Spider (Yu et al., 2018) by leveraging large language models. Specifically, we exploit ChatGPT's[2] text generation capabilities (`gpt-3.5-turbo-0301`) and ask it to rephrase Spider questions (no SQL- or database-specific information is provided; see Table 1), using three types of rewrite operations:

1. **Deletion** of words or phrases which are redundant for understanding the question's intent. For this purpose, we use two instructions: the first one *simplifies* the question, while the second one explicitly *hides unnecessary details* that do not change the meaning. The first instruction affords ChatGPT more freedom in rewriting the question. In Table 1, examples 1–2 show how Spider questions are reformulated with these instructions.
2. **Substitution** of words or phrases with simpler ones. We instruct ChatGPT to replace words with their *synonyms* and also to more generally attempt to *simplify by substituting a few words* in the question. In Table 1, examples 3–4 show how questions are rewritten with these instructions.
3. **Rewriting** of the entire question. Some questions can have the same meaning, despite being significantly dissimilar in their surface realisation. For example, the questions *Where do most people live?* and *Which cities have the largest population?* are related to the same database about cities and express the same intent but have no words in common. We instruct ChatGPT to provide *different ways of expressing* a question. We empirically find that ChatGPT can be too conservative at times and also include *a prompt with examples* to encourage more drastic reformulations. In Table 1, questions 5–6 show example outputs for these instructions.

We also ask ChatGPT to *paraphrase* questions (see example 7 in Table 1). This instruction may be viewed as a generalization of previous reformulations, however, in practice it is only somewhat helpful. ChatGPT often generates very similar versions of the original question, retaining the same details,

1. Instruction:	`Simplify`
Original Question	Which club has the most female students as their members? Give me the name of the club.
Output	Which club has the most female students?
2. Instruction:	`Simplify by hiding details`
Original Question	What is the title and credits of the course that is taught in the largest classroom (with the highest capacity)?
Output	What course is taught in the biggest classroom and what are its credits?
3. Instruction:	`Simplify using synonyms`
Original Question	What is the average duration in milliseconds of tracks that belong to Latin or Pop genre?
Output	What is the mean length in milliseconds of Latin or Pop songs?
4. Instruction:	`Simplify using substitutions`
Original Question	What are the locations that have gas stations owned by a company with a market value greater than 100?
Output	Where are the gas stations owned by a company worth more than 100?
5. Instruction:	`Express in a different way`
Original Question	What is the number of routes operated by the airline American Airlines whose destinations are in Italy?
Output	How many routes does American Airlines have that fly to Italy?
6. Instruction:	`Examples of the question simplification: <...>`
Original Question	What are the names of body builders whose total score is higher than 300?
Output	Who are the body builders with a score over 300?
7. Instruction:	`Paraphrase`
Original Question	Return the categories of music festivals that have the result "Awarded".
Output	List the categories of music festivals that have been recognized with awards.

Table 1: Different augmentations generated for Spider questions (see Appendix A, Table 6 for details).

style and structure following this instruction. In general, the approach advocated here rests on the assumption that the rewording of questions would *not* change their meaning, leading to an entirely different logical form. ChatGPT's conservative paraphrasing strategy is an advantage as almost all machine-generated questions preserve the meaning of the original question. We manually inspected 100 reformulations and found only 6% to be incorrect (i.e., inaccurate expressions of intent). Given this slight amount of noise, we use the generated questions without any filtering. Analysis in Appendix B further shows that our augmentations do

[2] chat.openai.com

not affect the nature of parsing errors.

5 Experimental Setup

Our experiments aim to evaluate the performance of models trained specifically for cross-database text-to-SQL parsing. We are interested in two types of generalization: robustness to controllable perturbations in utterances and adaptation to new domains with different question styles. Perturbations allow us to study more closely the impact of language variations, while new domains provide a more realistic and challenging setting. We first describe the datasets we use for training and evaluation and then briefly discuss the semantic parsing models we employ in our experiments.

5.1 Training Datasets

Our primary training dataset is Spider (Yu et al., 2018), which contains 7,000 questions to 140 different databases and 3,981 target queries (we exclude the single-domain datasets Yu et al. (2018) employ in addition to their data). Although there can be more than one question for the same intent (usually two), linguistic variations tend to be scanty and limited. We augment Spider with additional questions using ChatGPT as an automatic annotator. For each intent in the original training set, we generate two question reformulations based on the types specified in Section 4. We choose the augmentation types randomly and do not accept duplicates. The resulting augmented training set contains 14,954 instances; statistics for each category are in Table 2 and examples in Appendix E. The cost of calling the ChatGPT API to obtain our augmentations is approximately 7.5$.

5.2 Evaluation Datasets

The Spider development set consists of 1,034 questions to 20 databases and 564 target SQL queries. Since these questions share the same style and level of detail as the training set, we instead focus on evaluation sets with more natural and diverse language. Specifically, we focus on two groups of evaluation sets. The first group are datasets derived from the Spider development set, featuring identical SQL queries and databases which allow us to assess the model's resilience to variations in linguistic expression. The second group are independent datasets which not only differ in language usage but also in SQL style and database specifics. This setup enables us to evaluate model performance in more realistic conditions.

Augmentation Type	# examples
Simplify	774
Simplify by hiding details	1,136
Simplify using synonyms	1,285
Simplify using substitutions	1,316
Paraphrase	1,130
Express in a different way	1,065
Prompt with examples	1,256
Total	7,962

Table 2: Question reformulations generated for Spider; number of generations per instruction.

Datasets Based on Spider Chang et al. (2023) have recently released Dr.Spider, a comprehensive robustness benchmark which includes 9 evaluation sets with 7,593 examples of perturbations in natural language questions (NLQ sets). They have also created evaluation sets for database and SQL perturbations which are out of scope for this work. NLQ perturbation sets are based on the Spider development set, they contain the same databases and gold queries, deviating only in terms of the questions asked. They are generated with OPT (Zhang et al., 2022), a large pretrained language model, and manually filtered by SQL experts. There are three main categories of perturbations: change one or a few words that refer to SQL keywords (for example, replace the word *maximum* referring to the `max` SQL function with *the largest*), change references to columns (for example, replace *name of the countries* referring to column `CountryName` with *which countries*) and change references to database values (for example, replace *players from the USA* referring to the value `USA` with *American players*). Changes are made by replacing words with their synonyms or carrier phrases (e.g., *name of the countries* and *which countries*). Note that our augmentations target solely language variations and do not manipulate gold SQL queries.

Other Datasets GeoQuery (Zelle and Mooney, 1996) is a single-domain semantic parsing dataset with questions to a database of US geography. We use a version with SQL queries as logical forms and query-based splits (Finegan-Dollak et al., 2018) with a test set of 182 examples. GeoQuery questions are concise and their interpretation often depends on domain knowledge. For example, in the question *what is the largest city in the smallest state in the usa*, *the largest city* implies the city with the largest population but *the smallest state* implies the state with the smallest area.

KaggleDBQA (Lee et al., 2021) is a cross-domain text-to-SQL dataset for testing models under more realistic conditions. It contains 272 examples related to 8 real-world databases which can have abbreviated table and column names and "dirty" values. Questions were collected with annotators having access to column descriptions only, rather than the actual database schema (the dataset provides these descriptions but we do not use them). This simulates realistic database usage but also creates a challenge for semantic parsers as questions cannot be easily aligned to target SQL queries. For example, the question *Which artist/group is most productive?* to a database with information on hip hop torrents should be parsed into query `SELECT artist FROM torrents GROUP BY artist ORDER BY count(groupName) DESC LIMIT 1`, as *productive* refers to the number of releases and column `groupName` contains released titles.

5.3 Models

Current approaches frame text-to-SQL parsing as a sequence-to-sequence problem. The input is the concatenation of question and database entities, including table and column names, and content values extracted based on string matching, and the output is an SQL query. Shaw et al. (2021) show that a pre-trained T5-3B model (Raffel et al., 2020) fine-tuned on Spider (Yu et al., 2018) is a competitive text-to-SQL parser. Scholak et al. (2021) build on this approach with PICARD, a method for constrained decoding that filters the beam at each generation step, taking into account task-specific constraints such as grammatical correctness and consistency with the database. Recently, Li et al. (2023) propose RESDSQL, an approach that decouples schema linking from SQL parsing. They first filter relevant database entities and then use T5-3B to generate a sketch (i.e., SQL keywords) and then the actual SQL query. We use the best version of their model which also leverages NatSQL intermediate representations (Gan et al., 2021c).

We use the implementations from Scholak et al. (2021) and Li et al. (2023) for training models on augmented data and their released checkpoints for training on the original Spider. All models are trained for 100 epochs; we use a batch size of 200 for the base T5-3B to reduce the computational cost, leaving all other hyperparameters unchanged. We train on a single NVIDIA A100 GPU.

Our approach to data augmentation is model agnostic but our experiments focus on settings where the model is specifically trained or fine-tuned on text-to-SQL data. An alternative is large language models which are trained on huge text collections (including code) and able to translate natural language to SQL, without further fine-tuning on task-specific data (Rajkumar et al., 2022). Since our augmentations are generated by ChatGPT, a model trained with Reinforcement Learning for Human Feedback (Christiano et al., 2017), we include it as a standalone baseline. Following Liu et al. (2023), we prompt ChatGPT in a *zero-shot* setting with the description of the database schema followed by the question (the full prompt is shown in Appendix C). Large language models like ChatGPT differ from task-specific models in many respects, including potential use cases, resource requirements, transparency, and accessibility and thus any comparison should be interpreted with a grain of salt.

6 Results

Our experiments compare models trained on the original Spider data against models trained on augmented data. In addition, we report results for ChatGPT tested in a zero-shot mode. We evaluate model performance in two settings: zero-shot parsing on Spider-based data with *perturbed questions* and zero-shot parsing on *other datasets*. All results are evaluated with execution accuracy.

6.1 Robustness to Question Perturbations

Table 3 reports execution accuracy results on evaluation sets from Dr.Spider (Chang et al., 2023) which include perturbations in natural language questions. We also present results on the original Spider development set (see Appendix D for more results, including other Dr.Spider perturbation sets). Pre/Post refer to Spider subsets before/after perturbations (post-perturbation sets are the same subsets but with the questions rewritten).

We compare T5-3B with and without PICARD and RESDSQL models fine-tuned on the original Spider data and our augmentations; we also provide results for ChatGPT evaluated in the zero-shot setting. Our results show that ChatGPT is most vulnerable to question reformulations among all models. Chang et al. (2023) reach similar conclusions with Codex (Chen et al., 2021), another large pre-trained language model, and hypothesize this is due to the training data being biased towards docstrings (which is what most natural language

	T5-3B		Augmented T5-3B		PICARD		Augmented PICARD		RESDSQL		Augmented RESDSQL		ChatGPT	
Perturbation Set	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post
Keyword-synonym	70.2	62.6	73.8	65.4	72.6	66.3	75.3	69.4	81.5	72.4	**84.2**	**74.7**	64.7	55.7
Keyword-carrier	82.7	76.4	83.0	79.2	85.0	82.7	88.7	84.0	**89.0**	83.5	87.5	**85.0**	85.0	82.0
Column-synonym	63.9	51.3	66.3	54.2	71.0	57.2	68.7	**59.7**	**78.7**	63.1	77.4	**66.1**	66.1	48.8
Column-carrier	83.1	61.7	82.0	70.5	**86.9**	64.9	85.0	73.1	86.5	63.9	86.4	**76.3**	82.2	52.0
Column-attribute	49.6	48.7	60.5	58.8	58.8	56.3	63.9	62.2	**82.4**	**71.4**	**82.4**	**71.4**	77.3	62.2
Column-value	69.1	58.6	76.3	58.9	82.9	69.4	**83.2**	**70.4**	**96.4**	76.6	95.1	**77.6**	74.0	57.9
Value-synonym	68.6	46.4	68.6	53.0	72.5	53.0	70.8	**57.1**	79.2	53.2	**79.6**	55.1	69.0	45.8
Multitype	70.1	51.1	71.4	56.3	74.4	57.1	74.0	61.4	**83.8**	60.7	**83.8**	**65.7**	71.9	49.8
Others	75.3	73.1	76.6	72.7	79.6	**78.3**	**80.9**	77.6	**85.2**	79.0	84.8	**80.2**	74.0	66.4
Average	70.3	58.9	73.2	63.2	76.0	65.0	76.7	68.3	**84.7**	69.3	84.6	**72.5**	73.8	57.9
Spider Dev	74.4		75.3		79.3		79.3		**84.1**		84.0		72.2	

Table 3: Execution Accuracy on Spider development set and subsets taken from Dr.Spider (NLQ sets); model performance is shown before (Pre) and after perturbations (Post). We compare T5-3B, T5-3B+PICARD, and RESDSQL fine-tuned with and without augmentations and zero-shot ChatGPT.

		KaggleDBQA								
Model	GeoQuery	Nuclear	Crime	Pesticide	Math	Baseball	Fires	WhatCD	Soccer	Avg
T5-3B	54.4	59.4	48.2	16.0	7.1	20.5	43.2	7.3	16.7	27.3
+Augmented	60.4	56.3	48.2	18.0	7.1	20.5	**43.2**	**26.8**	22.2	30.3
PICARD	56.6	59.4	**51.9**	18.0	10.7	**25.6**	**43.2**	9.8	22.2	30.1
+Augmented	**62.6**	56.3	48.1	22.0	14.3	**25.6**	**43.2**	24.4	27.8	32.7
RESDSQL	56.6	59.4	48.1	16.0	**25.0**	23.1	**43.2**	17.1	22.2	31.8
+Augmented	59.3	**65.6**	44.4	**24.0**	**25.0**	23.1	**43.2**	19.5	**27.8**	**34.1**
ChatGPT	20.9	34.4	18.5	16.0	10.7	15.4	27.0	4.9	16.7	17.9

Table 4: Execution accuracy on GeoQuery test set (query splits) and different databases from KaggleDBQA. All models are tested in a zero-shot setting; +Augmented refers to models fine-tuned on the augmented Spider data.

utterances look like on websites like GitHub).

Execution accuracy for augmented models (T5-3B with and without PICARD and RESDSQL) improves by more than 3% compared to base models in almost all cases, while the accuracy gap on pre- and post-perturbed data decreases. Augmented RESDSQL delivers the highest post-perturbation accuracy of 72.5%. It also obtains the best results in almost all individual categories of post-perturbed sets confirming that our augmentations enhance robustness. Augmented models do not have an advantage over base models on the original Spider development set (see the last row in Table 3). There are two reasons for this: firstly, we augment questions only without adding new SQL queries, and secondly, augmentations shift the language distribution by removing specific details and rendering questions more natural, but the development set remains closer to the original training set.

6.2 Generalization to Other Datasets

Table 4 summarizes our results in the more challenging zero-shot setting. Specifically, we evaluate model performance on two out-of-domain datasets, namely GeoQuery (Zelle and Mooney, 1996) and KaggleDBQA (Lee et al., 2021). Both datasets differ from Spider in many respects, i.e., the types of questions being asked, the style of SQL queries, and the database structure.

We find ChatGPT performs very poorly on these datasets compared to models fine-tuned on Spider with or without augmentations. In all cases, augmented models improve execution accuracy compared to base models. PICARD trained with augmentations performs best on GeoQuery reaching an accuracy of 62.6% (a 6% difference against the base model). Augmented RESDSQL performs best on KaggleDBQA, which is more challenging, reaching an average accuracy of 34.1%. Augmenta-

Model	Spider Dev	Dr.Spider NLQ	GeoQuery	KaggleDBQA
T5-3B	74.4	58.9	54.4	27.3
+ Deletion	74.7	59.7	56.0	28.7
+ Substitution	75.1	62.9	56.0	**31.2**
+ Rewriting	75.0	62.3	53.8	27.4
+ Paraphrase	75.3	61.4	41.8	25.9
+ All (ours)	75.3	**63.2**	**60.4**	30.3
+ One Prompt	74.4	60.4	40.7	29.2
+ Spider-Syn	**75.6**	59.2	49.5	27.0
+ MT-TEQL*	75.0	62.0	47.8	29.2

Table 5: Execution accuracy on Spider development set, Dr.Spider NLQ sets, GeoQuery, and KaggleDBQA for T5-3B base and trained with different augmentations including Spider-Syn (Gan et al., 2021a) and sub-sampled (diacritic *) version of MT-TEQL (Ma and Wang, 2021).

tions are generally helpful but not across all individual categories (note that categories are represented by a limited number of examples per database and even a small number of errors can result in a drop of several percentage points). We suspect the low accuracy on KaggleDBQA is primarily due to challenges that are unrelated to language variation. In particular, its databases contain abbreviations which might be difficult to parse and SQL queries exemplify operations which are not present in Spider (e.g., arithmetic operators between columns).

6.3 Ablations and Analysis

We next investigate the impact of different types of question reformulations introduced in Section 4, and also compare against related augmentation methods: Gan et al. (2021a) manually annotate Spider-Syn with synonym substitutions, whereas Ma and Wang (2021) introduce MT-TEQL, a framework for generating semantics-preserving variants of utterances and database schemas. We use a version of MT-TEQL that changes prefixes and aggregator mentions in Spider questions. Additionally, we include a baseline which follows our procedure for data generation but uses only one prompt: provide *different ways of expressing* a question.

Table 5 shows the execution accuracy of T5-3B trained with and without augmentations pertaining to Deletion, Substitution, Rewriting, and Paraphrasing. We also include results with All augmentations combined. The ablation study shows that different types of augmentation are helpful for different datasets. On GeoQuery, models augmented with deletions and substitutions perform best; substitutions also perform best on the NLQ sets of Dr.Spider and KaggleDBQA. Paraphrasing-based augmentations are best for the original Spider development set, with Rewriting trailing behind. Results obtained with a single prompt (`express in a different way`) further illustrate the need for diverse instructions. We also trained T5-3B with augmentations from Spider-Syn (Gan et al., 2021a) and MT-TEQL (Ma and Wang, 2021). For a fair comparison, we randomly sample MT-TEQL examples with question transformations to match the training size obtained through our augmentations. As can be seen in Table 5, our combined augmentations outperform models trained on Spider-Syn and MT-TEQL on all evaluation sets (Dr.Spider NLG, GeoQuery, and KaggleDBQA).

The results in Table 5 reaffirm the observation that different evaluation sets exemplify different linguistic variations and that there is no single type of augmentation that represents them all. Rather, a *combination* of augmentations is needed to perform well *across* datasets. This in turn suggests that a model can acquire useful knowledge by being exposed to a *diverse* range of linguistic variations. We also observe that a model trained on combined augmentations outperforms models trained on more specialized datasets (i.e., Spider-Syn and MT-TEQL) which confirms that relying solely on local transformations of the questions is not sufficient for better generalization.

7 Conclusion

We propose to enhance the generalization capabilities of text-to-SQL parsers by increasing natural language variation in the training data. We leverage a large language model like ChatGPT to automatically generate a variety of question reformulations, thereby augmenting existing datasets with more natural and diverse questions. We evaluate state-of-the-art models trained with and without our augmentations on a variety of challenging datasets focusing on robustness (to perturbations) and out-of-domain generalization. Across models and datasets we find that augmentations improve performance by a wide margin. Our experiments further underscore the need for a broad range of augmentations representing the full spectrum of rewrite operations. In the future, we plan to explore the potential of large language models for multilingual semantic parsing.

Limitations

Our work aims to increase the robustness of semantic parsers against natural language variation but does not handle problems related to SQL queries and database structures that are also important for out-of-domain generalization. We obtain augmentations using ChatGPT, a black-box model provided by OpenAI, which limits its usage for non-academic purposes. Our augmentations are unfiltered and may add a small amount of noise to training data. Moreover, even though our proposed rewrite operations are diverse, they may still not cover all possible reformulations. In fact, we found it challenging for ChatGPT to generate wildly different expressions of the original intent. Finally, this work does not consider multilingual or conversational semantic parsing which we hope to explore in the future.

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 421–432, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 33st Annual Conference on Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S. Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 341–350, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. 2023. Dr.spider: A diagnostic evaluation benchmark towards text-to-SQL robustness. In *The 11th International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, Guss, et al. 2021. Evaluating large language models trained on code.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, volume 30, Long Beach, CA, USA. Curran Associates, Inc.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-SQL. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1337–1350, Online. Association for Computational Linguistics.

Kais Dukes. 2014. SemEval-2014 task 6: Supervised semantic parsing of robotic spatial commands. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 45–53, Dublin, Ireland. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. 2021a. Towards robustness of text-to-SQL models against synonym substitution. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

Processing (Volume 1: Long Papers), pages 2505–2515, Online. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021b. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8926–8931, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021c. Natural SQL: Making SQL easier to infer from natural language specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-SQL in the wild: A naturally-occurring dataset based on stack exchange data. In *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*, pages 77–87, Online. Association for Computational Linguistics.

Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2023. Annollm: Making large language models to be better crowdsourced annotators.

Shuo Huang, Zhuang Li, Lizhen Qu, and Lei Pan. 2021. On robustness of neural semantic parsers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3333–3342, Online. Association for Computational Linguistics.

Jiarong Jiang, Yiqun Hu, Wuwei Lan, Henry Zhu, Anuj Chauhan, Alexander Li, Lin Pan, Jun Wang, Chung-Wei Hang, Sheng Zhang, Marvin Dong, Joe Lilien, Patrick Ng, Zhiguo Wang, Vittorio Castelli, and Bing Xiang. 2022. Importance of synthesizing high-quality data for text-to-sql parsing. In *NeurIPS 2022 Workshop on SyntheticData4ML*.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, pages 13067–13075, Washington, DC, USA. AAAI Press.

Jingjing Li, Wenlu Wang, Wei-Shinn Ku, Yingtao Tian, and Haixun Wang. 2019. Spatialnli: A spatial domain natural language interface to databases using spatial comprehension. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '19, page 339–348, New York, NY, USA. Association for Computing Machinery.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability.

Pingchuan Ma and Shuai Wang. 2021. Mt-teql: Evaluating and augmenting neural nlidb on real-world linguistic and schema variations. *Proceedings of the VLDB Endowment*, 15(3):569–582.

Fatma Őzcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, and Vasilis Efthymiou. 2020. State of the art and open challenges in natural language interfaces to data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 2629–2636, New York, NY, USA. Association for Computing Machinery.

Xinyu Pi, Bing Wang, Yan Gao, Jiaqi Guo, Zhoujun Li, and Jian-Guang Lou. 2022. Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2007–2022, Dublin, Ireland. Association for Computational Linguistics.

Karthik Radhakrishnan, Arvind Srikantan, and Xi Victoria Lin. 2020. ColloQL: Robust text-to-SQL over search queries. In *Proceedings of the First Workshop on Interactive and Executable Semantic Parsing*, pages 34–45, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the*

59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 922–938, Online. Association for Computational Linguistics.

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

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to synthesize data for semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2760–2766, Online. Association for Computational Linguistics.

Kun Wu, Lijie Wang, Zhenghua Li, Ao Zhang, Xinyan Xiao, Hua Wu, Min Zhang, and Haifeng Wang. 2021. Data augmentation with hierarchical SQL-to-question generation for cross-domain text-to-SQL parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8974–8983, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*, volume 2, pages 1050–1055, Portland, Oregon. AAAI Press.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

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

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning.

A Data Generation

Table 6 shows the full versions of the prompts we use to generate the augmentations defined in Section 4 for the Spider training set.

1. Instruction:	`Simplify`
Full version	Simplify the following sentence: . . .
2. Instruction:	`Simplify by hiding details`
Full version	Simplify the sentence by hiding unnecessary details that do not change the meaning: . . .
3. Instruction:	`Simplify using synonyms`
Full version	Simplify the following sentence using synonyms: . . .
4. Instruction:	`Simplify using substitutions`
Full version	Make the sentence simpler by substituting some words in . . .
5. Instruction:	`Express in a different way`
Full version	What are different ways of expressing this question: . . .
6. Instruction:	`Examples of the question simplification: <. . .>`
Full version	Examples of the question simplification: Original: Find the names of stadiums whose capacity is smaller than the average capacity. Simplified: Which stadiums are smaller than the average? Original: Show the fleet series of aircraft flown by pilots younger than 34. Simplified: Return the fleet series of the planes whose captains are younger than 34. Original: Which cities have the largest population? Simplified: Where do most people live? Original: In which year was most of the ships built? Simplified: When were most of the ships constructed? Original: Tell me the number of orders with "Second time" as the order detail. Simplified: How many orders have "Second time" as an order detail? Original: . . . Simplified:
7. Instruction:	`Paraphrase`
Full version	Give me a paraphrase of the following question: . . .

Table 6: The full version of the prompts used for data generation.

Perturbation Set		T5-3B Pre	T5-3B Post	Augmented T5-3B Pre	Augmented T5-3B Post	PICARD Pre	PICARD Post	Augmented PICARD Pre	Augmented PICARD Post	RESDSQL Pre	RESDSQL Post	Augmented RESDSQL Pre	Augmented RESDSQL Post	ChatGPT Pre	ChatGPT Post
NLQ	Average	70.3	58.9	73.2	63.2	76.0	65.0	76.7	68.3	**84.7**	69.3	84.6	**72.5**	73.8	57.9
SQL	Comparison	62.9	62.4	71.3	66.3	68.0	68.0	74.2	70.8	80.9	82.0	**84.3**	**83.7**	73.6	64.0
	Sort-order	75.0	70.3	76.0	75.5	79.2	74.5	78.1	76.6	88.0	**85.4**	**88.5**	83.3	66.7	57.8
	NonDB-number	77.1	73.3	71.8	77.1	83.2	77.1	73.3	77.9	87.8	85.5	**90.8**	**90.8**	**90.8**	90.1
	DB-text	59.5	58.3	59.9	61.6	64.7	65.1	66.2	66.7	77.2	74.3	**91.5**	**75.0**	67.5	68.2
	DB-number	83.9	83.7	79.8	78.8	86.3	85.1	84.6	83.2	88.8	88.8	**91.5**	**91.2**	82.7	79.8
	Average	71.7	69.6	71.8	71.9	76.3	74.0	75.3	75.0	84.5	83.2	**89.3**	**84.8**	76.3	72.0
DB	Schema-synonym	66.4	46.9	67.8	52.8	73.0	56.5	73.4	61.9	**81.3**	68.3	80.9	**70.4**	67.6	56.0
	Schema-abbreviation	69.5	53.3	71.0	55.5	74.9	64.7	75.2	65.3	**82.4**	70.0	81.8	**71.7**	68.8	63.5
	Content-equivalence	84.6	40.8	72.3	46.1	88.7	43.7	86.9	37.2	90.3	40.1	**91.9**	41.4	81.2	**46.3**
	Average	73.5	47.0	72.3	46.1	78.9	55.0	78.5	54.8	84.7	59.5	**84.9**	**61.1**	72.5	55.3
All		71.3	59.9	72.6	62.7	76.6	65.9	76.6	67.9	84.7	71.7	**86.0**	**74.1**	74.3	61.5

Table 7: Execution Accuracy on subsets taken from Dr.Spider (NLQ, DB, and SQL sets); model performance is shown before (Pre) and after perturbations (Post). We compare T5-3B, T5-3B+PICARD, and RESDSQL fine-tuned with and without augmentations, and zero-shot ChatGPT.

B Error Analysis

In order to verify that our augmentations do not introduce new parsing errors, we examined examples in the Spider development set which were correctly parsed by a T5 model trained without augmentations but rendered incorrect after the same T5 model was trained with augmentations. Based on a sample of 60 instances, we observed that the majority of errors are similar in nature and symptomatic of a T5-trained semantic parser, e.g., errors in the output columns or join operation.

The only type of error that might be due to our augmentations concerns minor changes in values. Baseline T5 almost always copies values from the question but T5 trained with augmentations can slightly change them, e.g., use the full name instead of an abbreviation or lowercase instead of uppercase. We found this occurs in 10% of cases. Database values are mentioned verbatim in Spider questions but this could be different in real-world settings or other datasets where some tolerance to surface variations might be advantageous.

C ChatGPT Zero-Shot Prompt

Below we show the prompt we used when evaluating the zero-shot ChatGPT on text-to-SQL datasets following Liu et al. (2023):

```
### SQL tables, with their properties:
#
# stadium(Stadium_ID, Location, Name,
    Capacity, Highest, Lowest, Average)
# singer(Singer_ID, Name, Country,
    Song_Name, Song_release_year, Age,
    Is_male)
# concert(concert_ID, concert_Name,
    Theme, Stadium_ID, Year)
# singer_in_concert(concert_ID,
    Singer_ID)
#
### How many singers do we have? Return
    only a SQL query.
SELECT
```

D Additional Results

Table 7 shows our results on *all* Dr.Spider perturbation subsets (NLQ refers to subsets with perturbations in natural language questions, SQL and DB are perturbations in SQL and database tokens). We compare three models trained with and without augmentations: T5-3B, PICARD, and RESDSQL. We also employ ChatGPT in a zero-shot setting. Overall, the best model is augmented RESDSQL (74.1%) which is better than the base version by more than 2% on post-perturbed sets. Augmented T5-3B and PICARD also improve robustness compared to base models. Augmented RESDSQL delivers the best average results for all three types of perturbations and performs best on the majority of individual categories, even though our augmentations are *not* designed to improve robustness against SQL and DB perturbations.

Dataset	T5-3B	Augmented T5-3B	PICARD	Augmented PICARD	RESDSQL	Augmented RESDSQL	ChatGPT
Realistic	64.2	66.7	71.4	79.3	80.7	**84.0**	63.4
Spider-Syn	62.4	70.8	69.8	72.8	76.9	**79.2**	58.6
GeoQuery dev	59.1	64.2	64.2	**68.6**	59.7	54.1	25.8

Table 8: Execution accuracy on Spider-Realistic, Spider-Syn and GeoQuery dev set for T5-3B with and without PICARD and RESDSQL trained with or without augmentations.

Table 8 shows results on the additional evaluation sets, Spider-Realistic, (Gan et al., 2021a) Spider-Syn with 1,034 examples, and GeoQuery dev set with 152 examples (query splits of Finegan-Dollak et al. 2018). Both evaluation sets are based on the Spider development set, aiming to remove from the questions explicit references to database entities. These references were manually deleted or paraphrased in Spider-Realistic and replaced with synonyms in Spider-Syn. We observe that augmented RESDSQL obtains best results on both datasets (84.0% on Spider-Realistic and 79.2% on Spider-Syn) and is better than the base version by more than 4%. On the GeoQuery development set, the best model is augmented PICARD with 68.6% accuracy. Across *all* benchmarks, fine-tuned text-to-SQL parsers significantly outperform zero-shot ChatGPT.

E Examples of Spider Augmentations

We provide samples for the augmented Spider training set. Questions are grouped based on the intent. Types indicate whether the question is taken from the *original* Spider training set or is generated using one of the following instructions: *simplify*, simplify by *hiding details*, simplify by *synonyms*, simplify by *substitutions*, *express differently*, *paraphrase*, or by showing *examples*.

1. SQL query `SELECT personal_name FROM Students EXCEPT SELECT T1.personal_name FROM Students AS T1 JOIN Student_Course_Enrolment AS T2 ON T1.student_id = T2.student_id`

Questions		
	Find the personal names of students not enrolled in any course.	Type: original
	Which students not enrolled in any course? Find their personal names.	Type: original
	Find names of unenrolled students.	Type: hide-details
	Can you provide me with the names of the students who are not currently registered in any course?	Type: paraphrase

2. SQL query `SELECT donator_name , sum(amount) FROM endowment GROUP BY donator_name ORDER BY sum(amount) DESC`

Questions		
	List each donator name and the amount of endowment in descending order of the amount of endowment	Type: original
	Enumerate contributors and their endowment sums in decreasing order	Type: synonyms
	List donors and their endowments in descending order	Type: hide-details

3. SQL query `SELECT count(*) FROM CLASS AS T1 JOIN enroll AS T2 ON T1.class_code = T2.class_code WHERE T1.crs_code = 'ACCT-211'`

Questions		
	How many students enrolled in class ACCT-211?	Type: original
	What are the total number of students enrolled in ACCT-211?	Type: original
	How many pupils registered for course ACCT-211?	Type: synonyms
	How many students are enrolled in ACCT-211?	Type: from-examples

4. SQL query `SELECT T2.roomName FROM Reservations AS T1 JOIN Rooms AS T2 ON T1.Room = T2.RoomId WHERE firstname LIKE '%ROY%'`

Questions		
	Find the name of rooms booked by some customers whose first name contains ROY.	Type: original
	What are the name of rooms booked by customers whose first name has "ROY" in part?	Type: original
	What are the room names that have been reserved by customers with "ROY" in their first name?	Type: paraphrase
	What rooms did customers with "ROY" in their first name book?	Type: substitutions

5. SQL query `SELECT T1.account_name , T1.other_account_details FROM Accounts AS T1 JOIN Customers AS T2 ON T1.customer_id = T2.customer_id WHERE T2.customer_first_name = 'Meaghan' AND T2.customer_last_name = 'Keeling'`

Questions		
	Show the account name and other account detail for all accounts by the customer with first name Meaghan and last name Keeling.	Type: original
	What are the names and other details for accounts corresponding to the customer named Meaghan Keeling?	Type: original
	What are Meaghan Keeling's account names and details?	Type: simplify
	I am looking for the account details and names associated with Meaghan Keeling. Can you help me with that?	Type: paraphrase

6. SQL query `SELECT sum(acc_bal) FROM customer WHERE state = 'Utah' OR state = 'Texas'`

Questions		
	Find the total account balance of each customer from Utah or Texas.	Type: original
	What are the total account balances for each customer from Utah or Texas?	Type: original
	Add up the account balances of customers who live in Utah or Texas.	Type: express-differently
	What is the total account balance for customers from Utah or Texas?	Type: from-examples

7. SQL query `SELECT date_of_enrolment , date_of_completion FROM Student_Course_Enrolment`

Questions		
	List all the dates of enrollment and completion of students.	Type: original
	What are all the dates of enrollment and completion in record?	Type: original
	Provide a record of the enrollment and completion dates for all students.	Type: paraphrase
	What are the enrollment and completion dates of all students?	Type: from-examples

8. SQL query `SELECT headquarter FROM manufacturers WHERE founder = 'James'`

Questions		
	Where is the headquarter of the company founded by James?	Type: original
	What is the headquarter of the company whose founder is James?	Type: original
	Where was the company founded by James headquartered?	Type: express-differently
	Where is the main office of the company established by James?	Type: paraphrase

9. SQL query

```
SELECT max(Price) , max(Score) FROM WINE WHERE Appelation = 'St. Helena'
```

Questions	What are the maximum price and score of wines produced by St. Helena appelation?	Type: original
	Give the maximum price and score for wines produced in the appelation St. Helena.	Type: original
	What is the topmost price and score that can be obtained by wines produced in St. Helena?	Type: paraphrase
	What is the highest price and score for St. Helena wines?	Type: simplify

10. SQL query

```
SELECT degrees FROM campuses AS T1 JOIN degrees AS T2 ON t1.id = t2.campus
WHERE t1.campus = 'San Francisco State University' AND t2.year = 2001
```

Questions	What are the degrees conferred in "San Francisco State University" in 2001.	Type: original
	What degrees were conferred in San Francisco State University in the year 2001?	Type: original
	What diplomas were granted at SF State in 2001?	Type: synonyms
	What degrees were given at San Francisco State University in 2001?	Type: substitutions

11. SQL query

```
SELECT membership_card FROM member WHERE address = 'Hartford' INTERSECT
SELECT membership_card FROM member WHERE address = 'Waterbury'
```

Questions	What is the membership card held by both members living in Hartford and ones living in Waterbury address?	Type: original
	What is the membership card for people in Hartford and Waterbury called?	Type: substitutions
	Is there a membership card that is valid for both Hartford and Waterbury residents?	Type: express-differently

12. SQL query

```
SELECT kids FROM Reservations WHERE FirstName = 'ROY' AND LastName = 'SWEAZY'
```

Questions	How many kids stay in the rooms reserved by ROY SWEAZY?	Type: original
	Find the number of kids staying in the rooms reserved by a person called ROY SWEAZ.	Type: original
	How many children are staying in ROY SWEAZY's reserved rooms?	Type: from-examples
	How many kids are in Roy Sweaz's reserved rooms?	Type: hide-details

13. SQL query

```
SELECT count(*) FROM products AS t1 JOIN product_characteristics AS t2
ON t1.product_id = t2.product_id JOIN CHARACTERISTICS AS t3
ON t2.characteristic_id = t3.characteristic_id WHERE t1.product_name = 'laurel'
```

Questions	How many characteristics does the product named "laurel" have?	Type: original
	Count the number of characteristics of the product named 'laurel'.	Type: original
	How many features does "laurel" have?	Type: simplify
	How many qualities does the product "laurel" have?	Type: substitutions

14. SQL query

```
SELECT customer_name FROM customers WHERE payment_method = (SELECT payment_method
FROM customers GROUP BY payment_method ORDER BY count(*) DESC LIMIT 1)
```

Questions	What are the names of customers using the most popular payment method?	Type: original
	Find the name of the customers who use the most frequently used payment method.	Type: original
	Who are the customers using the popular payment method?	Type: hide-details
	Who are the customers utilizing the most favored payment option?	Type: synonyms

15. SQL query

```
SELECT TYPE FROM ship WHERE Tonnage > 6000 INTERSECT SELECT TYPE FROM ship
WHERE Tonnage < 4000
```

Questions	Show the types of ships that have both ships with tonnage larger than 6000 and ships with tonnage smaller than 4000.	Type: original
	What are the types of the ships that have both shiips with tonnage more than 6000 and those with tonnage less than 4000?	Type: original
	Display ships with tonnage above 6000 and below 4000.	Type: simplify
	Which types of ships have tonnage exceeding 6000 and also less than 4000?	Type: express-differently

16. SQL query

```
SELECT customer_name FROM customers EXCEPT SELECT t1.customer_name FROM customers AS t1
JOIN customer_addresses AS t2 ON t1.customer_id = t2.customer_id JOIN addresses AS t3
ON t2.address_id = t3.address_id WHERE t3.state_province_county = 'California'
```

Questions	Find the names of customers who are not living in the state of California	Type: original
	Discover the names of non-California customers.	Type: substitutions
	Who are the customers not residing in California?	Type: from-examples