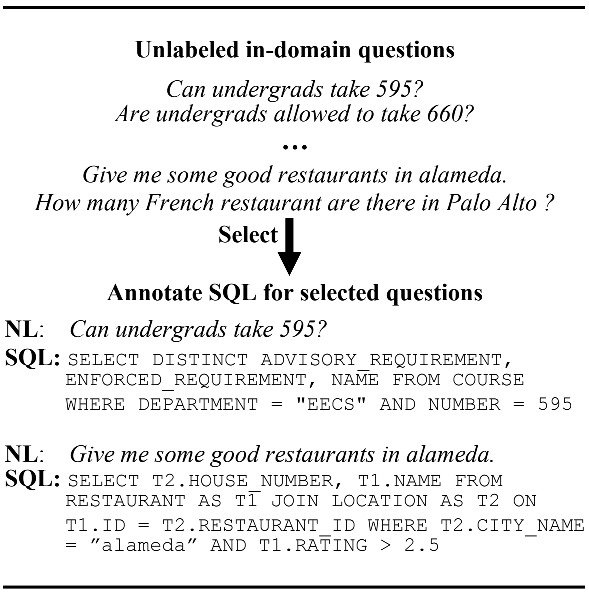
Exploring Example Selection for Few-shot Text-to-SQL Semantic Parsing

Anonymous ACL submission

Abstract

We study example selection methods for few-shot text-to-SQL tasks with unseen databases. Annotating natural language questions with corresponding SQL queries is expensive, but we can use abundant unlabeled questions to efficiently select examples to annotate and then use them to adapt models. Many previous works only randomly sample a few instances for few-shot learning, but this random selection is not sufficient to select representative and informative examples that provide specific domain knowledge. We thus explore methods to efficiently choose annotation examples. We identify two important factors: the diversity of selected instances and the dissimilarity to the source training data if any. A diverse training set contains more domain knowledge, while dissimilar examples are selected to fill in the domain gap between the source and target. We show that our best example selection approach substantially improves few-shot text-to-SQL performance in both *finetuning* using T5 and *in-context learning* with Codex: average execution accuracy gains of 8.7% and 4.3% over random selection. Our extensive analysis demonstrates the importance of the similarity metric and the embedding method for example representations. We also find that effective example selection reduces syntax errors on the target domains. Our results encourage future work to further explore example selection for efficient adaptation of text-to-SQL models.[1]



Unlabeled in-domain questions

Can undergrads take 595?
Are undergrads allowed to take 660?

...

Give me some good restaurants in alameda.
How many French restaurant are there in Palo Alto ?

Select

Annotate SQL for selected questions

NL: *Can undergrads take 595?*
SQL:
```
SELECT DISTINCT ADVISORY_REQUIREMENT,
ENFORCED_REQUIREMENT, NAME FROM COURSE
WHERE DEPARTMENT = "EECS" AND NUMBER = 595
```

NL: *Give me some good restaurants in alameda.*
SQL:
```
SELECT T2.HOUSE_NUMBER, T1.NAME FROM
RESTAURANT AS T1 JOIN LOCATION AS T2 ON
T1.ID = T2.RESTAURANT_ID WHERE T2.CITY_NAME
= "alameda" AND T1.RATING > 2.5
```

Figure 1: Our goal is to make the best use of the annotation budget by selecting a few examples to annotate from many unlabeled questions. The first two questions are similar, and it is enough to annotate only one of them. In the last two questions, we select the first one to annotate because it requires domain knowledge (e.g., *good restaurants* means the ones with *rating > 2.5*) to interpret its meaning.

1 Introduction

Text-to-SQL semantic parsing is the task of generating executable SQL queries from natural language utterances and relational database schemas. Most previous work aims to train and test a semantic parsing system on a single database (Price, 1990; Dahl et al., 1994; Zelle and Mooney, 1996a; Zettlemoyer and Collins, 2005; Dong and Lapata, 2016). However, it is inefficient to train a separate model from scratch for each possible target database, as there are too many in the world. Furthermore, annotating SQL queries for natural language questions requires annotators with technical backgrounds. Thus, much recent progress on this task has been driven by large-scale neural network models (Guo et al., 2019; Wang et al., 2020; Scholak et al., 2021) trained on cross-database semantic parsing (XSP) datasets such as Spider (Yu et al., 2018) that cover multiple databases and domains.

Nonetheless, these recent models trained on source domains (e.g., Spider) still perform poorly when applied to a different target domain (e.g., ATIS (Price, 1990)) with a wide variety of language usage not covered during training (Suhr et al., 2020). One major failure mode is caused by questions that require domain-specific knowledge to

[1]Our code is available at anonymized.

correctly interpret. For example, a user might ask “can undergrads take 595?” in an academic advising domain (Fig. 1). A text-to-SQL model struggles to interpret this question as “return the name and advisory and enforce requirements for Course 595 in the EECS department” unless it is trained on this particular domain.

We thus explore techniques to efficiently select examples to annotate for domain adaptation in text-to-SQL. In addition to clustering approaches that have proven successful in other tasks (Chang et al., 2021), we propose simple but effective sample selection methods for few-shot text-to-SQL. In particular, we focus on two important aspects as selection criteria: *example diversity* and *dissimilarity* to the source training data. We aim to select a few *diverse*, representative questions from the target domain and annotate corresponding SQL queries to maximize the limited annotation budget. For example, in Fig. 1 questions “Can undergrads take 595?” and “Are undergrads allowed to take 660” are similar, and models can learn from one of them. It is thus ideal to select one of the two instances to annotate and save annotation efforts. We also seek to select examples *dissimilar* to the source XSP training data (e.g., Spider) so that we encourage the model to learn domain-specific information not covered in the training data. For example, the last question in Fig. 1 is about restaurants, but understanding *good restaurants* requires domain knowledge (*rating > 2.5*), which differs from Spider whose questions usually specify such values.

We demonstrate the effectiveness of our selection methods on 9 datasets with varying domains under two settings: *finetuning* and *in-context learning*. In the former scenario, we first train a T5-large model (Raffel et al., 2020) on Spider and then finetune the model on a small number of selected in-domain question-SQL pairs. Our example selection boosts the Spider model’s few-shot performance on target domains by 8.7% in execution accuracy, as compared to random selection. In-context learning, on the other hand, constructs demonstration examples from the few annotated instances and feeds them to Codex (Chen et al., 2021), a variant of GPT-3 (Brown et al., 2020a) that is finetuned on publicly available code from Github. This approach has the advantage of avoiding the need for parameter updates of large language models. Our example selection is also effective in this setting with 4.3% accuracy improvement over random selection.

We also provide extensive analysis that examines the importance of the similarity metric and the embedding method used to produce representations of examples. In summary, our contributions are:

- We explore example selection methods that substantially improve the few-shot text-to-SQL performance on 9 diverse datasets.
- We apply our methods to finetuning and in-context learning, demonstrating the effectiveness in both settings.
- To the best of our knowledge, we are the first to develop a method for Codex in-context learning under a *small*, *predetermined* annotation budget. Our method consists of two stages where the first stage selects a few examples to annotate and the second stage further selects prompt demonstration examples from the annotated ones.
- We conducted detailed analyses on embedding methods and similarity metrics. We find that syntactic errors are substantially reduced by our example selection, compared to random selection.

2 Few-shot Text-to-SQL Approaches

We describe two major approaches to few-shot Text-to-SQL; each has its own strength, and we will demonstrate that our example selection is effective in both of them (§3). In both approaches, we first select a few examples to annotate. Those examples are then used in standard *finetuning* (§2.1) or prompt construction for *in-context learning* (§2.2).

2.1 Finetuning

Our finetuning proceeds over two steps: *source training* and *adaptation*. In source training, we finetune the pretrained T5-large encoder-decoder model (Raffel et al., 2020) for the Text-to-SQL task using the high-resource dataset of Spider (Yu et al., 2018). In the second step of adaptation, the model is further finetuned on M samples from the target domain (e.g., the ATIS flight booking domain, Price, 1990; Dahl et al., 1994). These M samples are selected based on the methods described in §3. We consider $M = 50$, and further explore the effects of increasing to $M = 250$ and $M = 500$. In both steps, the encoder-decoder model takes as input a concatenation of a natural language question and a string representation of the database schema (Hwang et al., 2019) and is trained to predict a corresponding SQL query. We train the model by minimizing the token-level cross entropy loss.

2.2 In-context Learning

In-context learning is a lightweight alternative to finetuning that keeps pretrained language model parameters frozen (Brown et al., 2020a). The language model takes as input a prompt that contains task descriptions, a few *demonstration* examples, and the input to be predicted on. While finetuning the whole network is a promising approach as discussed above, in-context learning has the advantage of avoiding finetuning on every target domain. This is particularly important for large-scale pretrained language models such as GPT-3, because finetuning is prohibitively expensive.

We introduce a method to apply in-context learning to our few-shot Text-to-SQL problem. We use Codex (Chen et al., 2021), a variant of GPT-3 finetuned on publicly available code from Github.[2] Similar to the finetuning setting, we first select and only annotate M natural language questions with gold SQL queries for the target domain ($M = 50, 250, 500$). Again similar to finetuning, we explore various selection methods that make crucial use of example diversity (§3). We then create a prompt for every evaluation instance by finding m demonstration examples most similar to the particular instance from the M annotated examples for *demonstration*; m is chosen to fit into the maximum length for Codex, and typically, $5 \leq m \leq 10$ (Zhao et al., 2021; Gao et al., 2021a). The similarity of a pair of examples is measured by their cosine distance in the T5 embeddings with averaging pooling over all encoder time steps and normalization by subtracting the mean of all input embeddings.

3 Selection Methods

In few-shot text-to-SQL, we assume access to a large set of unlabeled user questions on the target domain. This simulates many real-world scenarios where we have a record of many natural language questions from customers, but not their SQL annotations. Our goal is to select a small subset of examples to be annotated with SQL queries that will be used for few-shot finetuning (§2.1) and in-context learning (§2.2). We hypothesize that effective examples should be *diverse* to represent different types of user questions. To this end, we propose two types of example selection strategies. The first strategy (§3.1) employs a sequential selection process by iteratively picking examples that are dissimilar to the already selected ones. Furthermore, for the sequential selection strategy in our finetuning setup (§2.1), we also prioritize selecting the target domain examples that are dissimilar to the source domain examples so that the model can be better adapted using more domain-specific information. The second strategy (§3.2) clusters the user questions into groups of similar examples and then selects the example closest to the centroid from each cluster as the representative.

3.1 Sequential Selection based on Similarity

Our first strategy is sequential selection by picking examples iteratively while making sure they are dissimilar to each other to promote sample diversity. We propose the following two methods of sequential selection. **SelfDis** (self-dissimilar) uses dissimilarity with the already chosen examples to choose the next one. In the finetuning setup where the model is first trained on the source domain, **SrcSelfDis** (source-training-data-self-dissimilar) further incorporates dissimilarity to the source domain examples .

SelfDis Let $\mathcal{D}_{\text{sel}}$ denote the set of selected examples. We randomly select a seed example from the training set to initialize $\mathcal{D}_{\text{sel}}$, and we iteratively add new examples. For each remaining training example $\boldsymbol{x}_i$, we compute the dissimilarity between $\boldsymbol{x}_i$ and the already chosen examples $\boldsymbol{x}_j$ in $\mathcal{D}_{\text{sel}}$.

$$\text{SelfDis}(\boldsymbol{x}_i) = -\sum_{\boldsymbol{x}_j \in \mathcal{D}_{\text{sel}}} \cos(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

At each iteration, We choose the example with the highest SelfDis score such that it is the farthest from $\mathcal{D}_{\text{sel}}$ on average. We stop iteration when $\mathcal{D}_{\text{sel}}$ includes M examples. The final $\mathcal{D}_{\text{sel}}$ is expected to be a diverse subset of the unlabeled (i.e., natural language questions without SQL annotations) training set.

SrcSelfDis When the model is first trained on the source domain, we want to select target domain examples that are dissimilar to the source training examples for domain adaptation. Therefore, we propose SrcSelfDis that combines the similarity to the examples in the source domain and that to the already selected examples. Let $\mathcal{D}_{\text{src}}$ denote the set of source domain training examples. We first compute the **SrcDis** (source-training-data-dissimilar) score for dissimilarity to the source domain examples.

$$\text{SrcDis}(\boldsymbol{x}_i) = -\sum_{\boldsymbol{x}_j \in \mathcal{D}_{\text{src}}} \cos(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

[2] https://openai.com/blog/openai-codex/

	GeoQuery	Advising	ATIS	Yelp	Scholar	Restaurants	IMDB	Academic	Kaggle	Spider
Database	1	1	1	1	1	1	1	1	8	160
Train	576	2756	15786	99	1139	266	102	141	272	7000
Evaluation	344	242	1757	42	252	112	45	59	185	1034

Table 1: Numbers of databases, train and evaluation instances in each dataset. We follow the dataset format in Finegan-Dollak et al. (2018a), but report the number of question-SQL pairs instead of the separate question and SQL query counts for the first 8 datasets.

Then, we add SelfDis and SrcDis as the SrcSelfDis score using α as the scaling parameter to balance the two terms. We use $\alpha = 1700$ in the later experiments.

$$\text{SrcSelfDis}(\boldsymbol{x}_i) = \text{SrcDis}(\boldsymbol{x}_i) + \alpha\text{SelfDis}(\boldsymbol{x}_i)$$

The first instance is chosen as the one with the maximum SrcDis score, and we sequentially add examples with the maximum SrcSelfDis score until M examples are selected.

3.2 Selection by Clustering

Alternatively, we can cluster the whole unlabeled dataset into M groups. Each cluster represents a group of similar examples. Therefore, we select one instance from each cluster. Since the example closest to the cluster centroid minimizes the total distance to the remaining examples in the same cluster, we consider it as the representative of this cluster. By selecting all cluster representatives, we aim to maximize the diversity of the selected subset. In addition, the model prediction of an evaluation instance usually depends on the nearest neighbor in the training set (Khandelwal et al., 2019; Kwon et al., 2021). By diversifying the selected instances, we maximize the chance for an evaluation data point to find a similar example in the few-shot finetuning set. We consider the following two clustering algorithms.

K-means K-means groups unlabeled training instances into M clusters based on their embedding representations. We then choose the example closet to the centroid of each cluster.

Agglomerative Agglomerative clustering iteratively merges the closest two clusters. It constructs a hierarchy of unlabeled instances, where clusters on a higher level are more dissimilar to each other. We terminate the Agglomerative clustering algorithm when only M clusters are left. Similar to K-means, we then choose the example closet to the centroid of each cluster.

4 Experiments

In this section, we first discuss our experimental setups (§4.1). We then describe the experimental results and compare varying selection methods extensively (§4.2).

4.1 Experimental Setup

Datasets Following Suhr et al. (2020), we use Spider (Yu et al., 2018) as the source training dataset and 8 other single-domain datasets to perform few-shot domain adaptation: GeoQuery (Zelle and Mooney, 1996a), Advising (Finegan-Dollak et al., 2018a), ATIS (Price, 1990; Dahl et al., 1994), Scholar (Iyer et al., 2017), Restaurants (Tang and Mooney, 2000; Popescu et al., 2003; Giordani and Moschitti, 2012), Academic (Li and Jagadish, 2014), Yelp (Yaghmazadeh et al., 2017), and IMDB (Yaghmazadeh et al., 2017). In addition, we further evaluate our methods on the Kaggle dataset (Lee et al., 2021) containing multiple databases. We follow the the standard splits of train and evaluation in the datasets if they have, and randomly split datasets into 70% for training and 30% for evaluation otherwise. The dataset information is summarized in Table 1; evaluation examples in each dataset are available in Appendix E.

Evaluation Metrics Following Zhong et al. (2020), we use the *test suite accuracy* for all datasets as evaluation metrics. Instead of using a single given database to compute execution accuracy, it compares the *execution results* of the predicted queries and the gold queries on a compact test suite of databases with designed instances to distinguish different values in each clause. This setting reduces false positives of traditional execution accuracy (i.e., wrong SQL queries but happen to have the same execution result as correct ones). The test suite of databases are generated by modifying one aspect of the gold queries, and therefore, gives a tight upper bound compared to other evaluation metrics such as the exact set match (Yu et al., 2018) and execution accuracy based on a small database.

Implementation We implement two clustering-based selection methods using scikit-learn pack-

Method	GeoQuery	Advising	ATIS	Scholar	Academic	Restaurants	Yelp	IMDB	Kaggle
				Finetuning (T5-Spider)					
Random	$58.8_{3.8}$	$16.6_{6.8}$	$34.2_{5.0}$	$62.4_{5.7}$	$40.3_{3.7}$	$93.7_{2.7}$	$45.6_{8.2}$	$42.4_{7.7}$	$34.5_{4.0}$
K-means	$64.5_{2.8}$	$22.4_{4.4}$	$38.2_{2.1}$	$66.8_{3.0}$	$47.6_{2.2}$	$96.6_{1.2}$	$55.0_{5.7}$	$45.7_{4.2}$	$36.8_{2.3}$
Agglo	66.9	25.1	39.0	**75.6**	49.2	98.2	57.1	48.9	35.2
SelfDis	$65.5_{3.3}$	$24.4_{5.3}$	$38.8_{2.5}$	$69.8_{4.1}$	$48.2_{2.7}$	$97.4_{1.8}$	$55.7_{6.3}$	$46.6_{5.6}$	$35.7_{3.2}$
SrcSelfDis	**68.0**	**26.4**	**40.6**	73.8	**50.9**	**99.1**	**59.5**	**51.1**	**37.8**
				In-context Learning (Codex)					
Random	$57.9_{2.1}$	$16.2_{3.2}$	$31.6_{2.3}$	$64.3_{4.7}$	$51.2_{3.4}$	$90.2_{1.8}$	$33.3_{5.6}$	$41.2_{4.5}$	$2.4_{1.0}$
K-means	$62.4_{0.8}$	$18.3_{2.4}$	$34.5_{1.4}$	$63.2_{2.4}$	$54.8_{1.8}$	$90.7_{0.5}$	$34.5_{2.6}$	$42.5_{2.7}$	$2.8_{0.6}$
Agglo	63.5	18.4	35.2	**65.2**	55.6	91.2	36.7	42.8	3.2
SelfDis	$\mathbf{65.4}_{1.1}$	$\mathbf{20.6}_{2.6}$	$\mathbf{36.7}_{2.2}$	$63.5_{2.4}$	$\mathbf{58.9}_{2.3}$	$\mathbf{91.4}_{0.9}$	$\mathbf{41.3}_{2.3}$	$\mathbf{43.6}_{3.3}$	$\mathbf{5.2}_{0.8}$

Table 2: Finetuning (T5-Spider) and in-context learning (Codex) performance with different selection methods. All scores are based on the test suite accuracy (§4.1). 50 examples are selected to annotate. Among the five selection methods for finetuning, SrcSelfDis performs best apart from Scholar (8.7% improvement on avg. compared to Random). For in-context learning, SelfDis improves the average accuracy by 4.3% and reduces the variance. The subscripts indicate standard deviations from 6 trials for the three methods involving random sampling (Random, K-means, and SelfDis). Bold numbers indicate the best results in finetuning or in-context learning.

ages.[3] We use the T5-large model checkpoint trained on Spider from Scholak et al. (2021) which achieves 65.3% exact matching accuracy. For simplicity, we refer to T5-large and T5-large trained on Spider as T5 and T5-Spider respectively. In all our experiments, the batch size per device is set to 2 with gradient accumulation steps of 2. We use greedy search (i.e., beam size 1) since we found that a larger beam size did not improve the performance. We finetune the T5-Spider on in-domain selected examples of each evaluation dataset for 30 epochs. Following Shaw et al. (2020) and Scholak et al. (2021), for the input sequence in finetuning, we also serialize the database schema as a string and append it to the question. For in-context learning, we use the Davinci version of Codex; the input sequence is the concatenation of task descriptions, demonstration examples and a question *without* the database schema, as more examples can be added under the maximum input length in this case.[4]

4.2 Results

We evaluate the performance of our selection methods on the aforementioned datasets. As shown in Table 2 where $M = 50$ in-domain examples are selected, all selection selection methods (K-means, SelfDis, Agglo, and SrcSelfDis) show substantial improvements over random selection both in the finetuning and in-context learning experiments. This confirms our hypothesis that random selection is suboptimal. K-means, SelfDis and Agglo can greatly outperform the random baseline. Even with random initial states, K-means and SelfDis have smaller variance and are more stable in selecting representative instances. Finally, SrcSelfDis performs best among all five methods in all datasets except Scholar. Overall, our combined approach–using both the diversity and dissimilarity to source training data as criteria–achieves a 8.7% performance gain on average.

Moreover, K-means, Agglo and SelfDis give substantial improvements on in-context learning. The diversity in the M annotated examples can help us find similar m demonstration examples to every evaluation instance when its prompt is created; when the annotated examples are more homogeneous, it is possible that we cannot find similar demonstration examples to some evaluation instances, resulting in performance degradation.

In general, since questions in GeoQuery, Scholar and Restaurants contain less unspecified domain knowledge (as mentioned in §1), T5-Spider achieves higher performance compared to other datasets. We also found that our best method SrcSelfDis can select more question and SQL query templates (see more details in Appendix B).

Comparison between T5 and Codex As shown in Table 2, our example selection is effective in both settings, but T5 generally outperforms Codex when random, K-means, Agglo or SelfDis selection is applied. The only exception is Academic, where Codex outperforms T5 by a large margin. A potential reason is that, in both training and evaluation

[3] https://scikit-learn.org/stable/modules/clustering.html.

[4] We tried adding serialized schemas in the same way as finetuning, but we found that it did not improve performance.

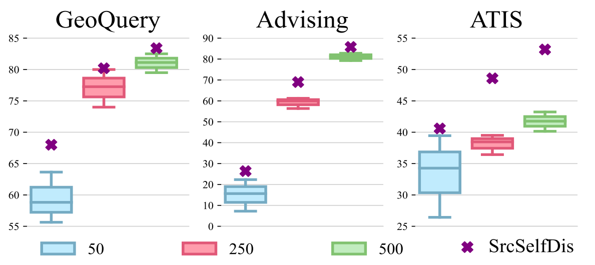


Figure 2: Effects of annotation sizes (i.e., number of selected examples) for the finetuning setting. Boxplots are random selection. The purple cross is SrcSelfDis selection. With different in-domain training data size, SrcSelfDis consistently improves the model performance. Exact numbers in Table 7 of Appendix D.

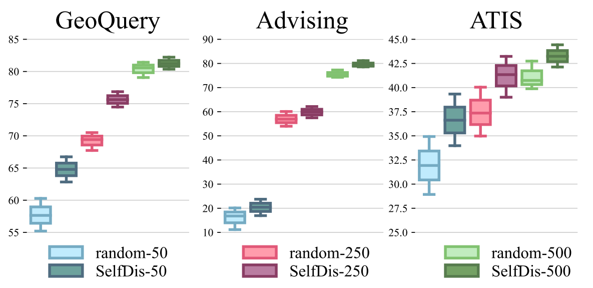


Figure 3: Effects of annotation sizes (i.e., number of selected examples) for the in-context learning setting. SelfDis selection in the first phase improves the model performance and reduces the variance across different annotation sizes. See Table 8 in Appendix D for more detailed results.

splits of the Academic dataset, questions always start with "return me". The consistent question patterns might particularly benefit Codex, which learns domain-specific knowledge directly from the demonstration examples in the prompt.

Codex lags behind T5 most in Yelp and Kaggle. Since Yelp has a significantly larger portion of templates with only one concrete examples, in the prompt construction for in-context learning, it is much more difficult to select similar instances to each test example. As for Kaggle, it is a dataset containing multiple databases, so it is difficult for the model to effectively learn much domain knowledge without the serialized schema or similar examples in the prompt. This explains the performance degradation of Codex in Yelp and Kaggle.

5 Analysis

In this section, we analyze our example selection methods along three dimensions: effects of annotation sizes, embedding methods for example representations, and the similarity metrics. We then provide an error analysis to better understand the improvement gained from our methods. For simplicity, we focus on three representative datasets with various sizes and complexity: GeoQuery, Advising, and ATIS. We found similar results from the other datasets.

5.1 Effects of Annotation Sizes

As shown in Fig. 2, SrcSelfDis consistently improves the model performance when different numbers of labeled instances are available. In particular, it achieves the most substantial improvement when only a small number of examples are annotated. For example, the model achieves the performance gain of 9.8% ($M = 50$), 9.4% ($M = 250$), and 4.5% ($M = 500$) on Advising. The largest improvement from $M = 50$ indicates that our approach is particularly suitable when annotation budgets are limited. Similar patterns are also observed in the in-context learning, as shown in Fig. 3. Both in finetuning and in-context learning, the model performance variance decreases when the training set is larger. Another interesting finding is that, in the ATIS dataset, when 250 examples are selected by SelfDis, the model performance surpasses that when 500 examples are selected randomly. This indicates that SelfDis helps the model achieve better performance under lower annotation and computational budgets.

Comparison between T5 and Codex Combining the results from Figs. 2 and 3 (See more details on Table 7 and 8 in Appendix), we can see that the performance differences between T5 and Codex are the largest when the annotated examples are of medium size. For example, in GeoQuery, when T5 uses SrcSelfDis and Codex uses SelfDis to select examples, their performance differences are 2.6%, 4.8% and 2.2% when $M = 50, 250, 500$ respectively. The potential reason is that, very few examples cannot cover sufficient domain knowledge for the models to learn, while many examples provide the models with abundant information. This explains the relatively small differences when the annotated examples are very limited ($M = 50$) or abundant ($M = 500$). In contrast, when the labeled instances are of medium size ($M = 250$), the gap between the two settings is large, which implies that T5 captures additional domain information more effectively when M increases from 50 to 250. Although this observation is not well aligned with the ATIS dataset, it is probably due to its much larger size than GeoQuery and Advis-

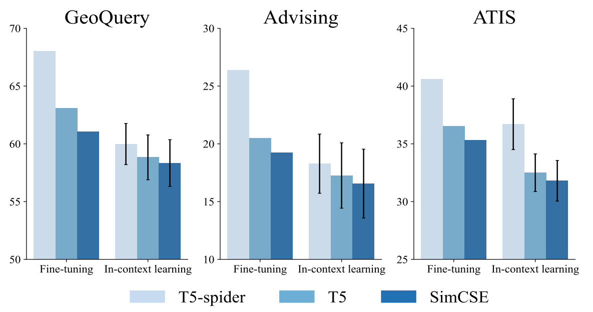


Figure 4: Comparison of three embedding methods (T5-Spider, T5, and SimCSE) in the finetuning and in-context learning settings over three domains. T5 indicates embeddings obtained from off-the-shelf T5 without any finetuning. In all domains, T5-Spider performs best. See Table 9 in Appendix D for more detailed results.

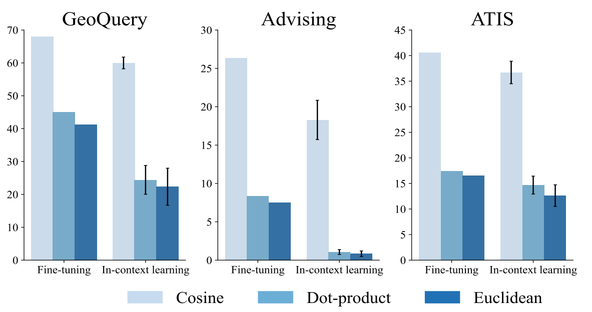


Figure 5: Comparison of three similarity metrics (cosine similarity, dot-product, and Euclidean distance) in the finetuning and in-context learning settings over three domains. In all domains, Cosine similarity performs the best. See Table 10 in Appendix D for more detailed results.

ing (Table 1). In this case, 500 examples are still considered as medium size for ATIS.

In summary, with the benefit of avoiding finetuning, it may be better to use Codex when we have very abundant or very few labeled instances, given the small performance difference from T5 finetuning. However, when the labeled instances are of medium size, it is more important to consider the tradeoff between finetuning T5 and its performance gain over Codex, in addition to its accessibility and the large model size. Note that future advances in prompt engineering can push the Codex performance.

5.2 Design Choices for Example Selection

Embedding methods Fig. 4 compares finetuning and in-context learning results from three embedding methods: T5-Spider (our default, T5 finetuned on Spider), T5 (T5 without any finetuning), and the SimCSE model (Gao et al., 2021b) trained on unlabeled instances. For fair comparisons, we use SrcSelfDis to select examples in finetuning and SelfDis in in-context learning for all the embedding methods. T5-Spider is consistently best across three datasets. This illustrates the effectiveness of T5 finetuning on Spider for embedding learning and thus sample selection.

Similarity metrics We then use the default embeddings from T5-Spider and compare three similarity metrics for SrcSelfDis and SelfDis. Fig. 5 shows that in both settings, when the cosine similarity is applied in the selection process, the model consistently performs better than Euclidean distance and dot product. This indicates the importance of similarity metrics for the success of SrcSelfDis and SelfDis.

	model	GeoQuery	Advising	ATIS
Random	T5	20.2	47.6	27.3
SrcSelfDis	T5	16.4	40.7	23.2
Random	Codex	19.64	18.34	16.7
SelfDis	Codex	18.96	17.86	15.2

Table 3: Syntax error rates over 50 examples in three domains. We compare random, SrcSelfDis for finetuning (T5), and SelfDis for in-context learning (Codex). With SrcSelfDis selection for T5 or SelfDis for Codex, the syntax error rate is substantially reduced.

5.3 Error Analysis

We consider two major error types: *syntax error* and *semantic error*. Syntax errors result in a query execution failure, while semantic errors lead to wrong execution results. We randomly select 50 examples from each of three domains and calculate the percentage of wrong predictions caused by syntax errors.

The results are in Table 3. When T5 and Codex select examples by SrcSelfDis and SelfDis respectively, the syntax error rate is substantially reduced compared to the random baseline. This suggests that diverse instances selected by SrcSelfDis and SelfDis contribute to the model capacities of writing syntactically correct SQL queries in different domains with various question styles. For example, in the Advising dataset, the model trained on examples selected by SrcSelfDis predicts correctly on the question, "of the upper-level courses, are any taught by deorio?"; in contrast, the model trained on randomly-selected examples does not. Indeed, SrcSelfDis selected an example with a similar question: "of the upper level courses, which does Artan teach," which corresponds to the same SQL query template. In the in-context learning, we also observe a reduction of the syntax error rate when SelfDis is applied.

Furthermore, SQL annotation style divergence between source training and target finetuning substantially decreases the model performance in T5-Spider finetuning, but not much in Codex in-context learning, as Codex is not trained on Spider (see more details in Appendix A).

6 Related Work

Text-to-SQL Semantic Parsing Semantic parsing has been researched for decades (Zelle and Mooney, 1996b; Miller et al., 1996; Zettlemoyer and Collins, 2005; Pasupat and Liang, 2015; Dong and Lapata, 2016). Among various logical forms, Text-to-SQL has attracted much interest for its practical usefulness (Zhong et al., 2017; Yu et al., 2018; Finegan-Dollak et al., 2018b; Bogin et al., 2019; Wang et al., 2020). While they mainly work with full access to annotated training data, we study few-shot learning for Text-to-SQL for both domain adaptation finetuning and GPT-3 in-context learning. Shin et al. (2021) recently proposed to map natural utterances into canonical formats that can be automatically converted to a target meaning representation and showed its effectiveness with large pretrained language models in a few-shot setting. In this work, we focus particularly on example selection for few-shot learning.

Domain Adaptation for Semantic Parsing Several recent papers focus on adapting semantic parsers trained on source domains to target domains (Li et al., 2020; Chen et al., 2020), including Text-to-SQL (Suhr et al., 2020; Wang et al., 2021). Suhr et al. (2020) test a model performing well on Spider on another 8 Text-to-SQL datasets in varying domains, and the model does not generalize well. Prior work (Suhr et al., 2020; Gan et al., 2021) find that the model fails to generalize because of domain-specific phrases, diverging database and query structures, dataset conventions, and the challenges of identifying entities in natural utterances and mapping entities to database columns. Wang et al. (2021) use meta-learning to better generalize models to different domains on Spider, where the model is optimized for the source and target domain performance simultaneously. In contrast to these works, we study example selection in few-shot learning for efficiently adapting models to low-resource target domains.

Example Selection in Few-shot Learning Few-shot learning for NLP has received increasing interest with the emergence of prompt-based methods (Schick and Schütze, 2020) and large pretrained language models (Raffel et al., 2020; Brown et al., 2020b). Empirical results of few-shot learning heavily depend on the choice of training examples. Liu et al. (2021) retrieve examples that are semantically similar to a test sample to formulate its prompt for GPT-3. Shin et al. (2021) use several pretrained language models as few-shot semantic parsers with GPT-3 for selecting the most relevant training examples to create prompts. Chang et al. (2021) study the effects of K-means example selection strategies on data-to-text generation, document summarization, and question generation. Our work also follows the line of work on unsupervised subset selection where labels are not used for selecting examples (Har-Peled and Mazumdar, 2004; Wei et al., 2014, 2015; Karamcheti et al., 2021). Furthermore, active learning has also been used for dynamically acquiring new labeled examples during the training process based on uncertainty (Thompson et al., 1999; Kasai et al., 2019; Yao et al., 2020), entropy (Siddhant and Lipton, 2018; Gal et al., 2017) and etc. In contrast, we study comprehensive example selection methods including both clustering and sequential selection based on similarity. Compared with active learning, our example selection approach is more efficient because we perform example selection before training without updating and querying models iteratively.

7 Conclusion

We explored several example selection methods for Text-to-SQL that can be applied to the traditional few-shot learning via finetuning and prompt construction of Codex (GPT-3) in-context learning. We identified the diversity and the dissimilarity to source training data as two important factors for selection. We showed that with carefully-chosen annotated instances, the model performance on the 9 target domains can be greatly enhanced. Our further analyses examined the influence of embeddings and similarity metrics as well as the change in syntax error rate. Our simple yet effective methods are easy to implement. We plan to apply them to other tasks as future work.

References

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-sql parsing. In *EMNLP*.

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. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

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

Ernie Chang, Xiaoyu Shen, Hui-Syuan Yeh, and Vera Demberg. 2021. On training instance selection for few-shot neural text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 8–13.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv e-prints*, pages arXiv–2107.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. *arXiv preprint arXiv:2010.03546*.

Deborah A Dahl, Madeleine Bates, Michael K Brown, William M Fisher, Kate Hunicke-Smith, David S Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018a. 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.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018b. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021. Exploring underexplored limitations of cross-domain text-to-sql generalization. In *EMNLP*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Alessandra Giordani and Alessandro Moschitti. 2012. Automatic generation and reranking of sql-derived answers to nl questions. In *International Workshop on Eternal Systems*, pages 59–76. Springer.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and D. Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *ACL*.

Sariel Har-Peled and Soham Mazumdar. 2004. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300.

Wonseok Hwang, Ji-Yoon Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. In *KR2ML Workshop at NeurIPS*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher D Manning. 2021. Mind your outliers! investigating the negative impact of outliers on active learning for visual question answering. *arXiv preprint arXiv:2107.02331*.

Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Yongchan Kwon, Manuel A Rivas, and James Zou. 2021. Efficient computation and analysis of distributional shapley values. In *International Conference on Artificial Intelligence and Statistics*, pages 793–801. PMLR.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. Kaggledbqa: Realistic evaluation of text-to-sql parsers. *arXiv preprint arXiv:2106.11455*.

Fei Li and Hosagrahar V Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Zechang Li, Yuxuan Lai, Yansong Feng, and Dongyan Zhao. 2020. Domain adaptation for semantic parsing.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for GPT-3? *ArXiv*.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. 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)*.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157.

Patti Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

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*.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

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.

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

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. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Aditya Siddhant and Zachary C Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909.

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*. Association for Computational Linguistics.

Lappoon R Tang and Raymond Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *ICML*.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Meta-learning for domain generalization in semantic parsing. *ArXiv*, abs/2010.11988.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *ACL*.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2015. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963. PMLR.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. 2014. Unsupervised submodular subset selection for speech data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4107–4111. IEEE.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26.

Ziyu Yao, Yiqi Tang, Wen tau Yih, Huan Sun, and Yu Su. 2020. An imitation game for learning semantic parsers from user interaction. In *EMNLP*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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.

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

John M. Zelle and Raymond J. Mooney. 1996b. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR. AAAI Press/MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *International Conference on Machine Learning (ICML)*.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suite. In *The 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Effects of SQL Annotation Styles

The query styles in 9 downstream datasets are different from those in Spider. For example, instead of always using "!=", the queries in Advising use "<>" to get the same output. We examine the effects of different query styles on finetuning the model. By applying the same selection method SrcSelfDis, we select the same set of training instances without rewriting queries. Fig. 6 shows that queries having styles that the models are unfamiliar with result in significantly worse performance in the finetuning. Since test suite accuracy compares query execution results, surface style differences are ignored during evaluation. This suggests that the finetuning model benefits from the consistency of query styles in source training, finetuning, and evaluations. However, since Codex has no source training stage, whether the queries follow the styles in Spider or not has little influence on the performance.[5]

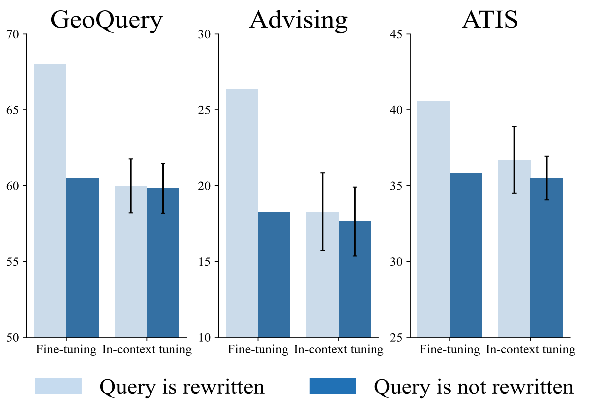


Figure 6: We use SrcSelfDis(SelfDis) to select 50 examples for T5(Codex), and compare the T5(Codex) performance between: 1) the SQL queries in the selected examples are rewritten to follow query styles in Spider; 2) the SQL queries are not rewritten. The finetuning performance drops significantly when if the query is not rewritten, while the performance of Codex only drops a little. Detailed results can be found in Table 11.

B Selection Quality

We also use the question and SQL query templates provided by Finegan-Dollak et al. (2018a) to conduct our analysis. The question and SQL query templates omit specific values but preserve sentence structures with patterns and styles. For example, "What is the biggest city in state_name0" is a question template, where "state_name0" can be the name of any state in the United States. By substituting different values in the templates, different question-SQL pairs are generated. We confirm that SrcSelfDis selects more diverse examples by calculating the number of templates in the selected examples. As questions and SQL queries from the same template are highly similar, we consider the number of question and SQL query templates as a diversity measurement for the selected instances. As shown in Table 4, more question and SQL tem-

[5]Instead of no inluence, since the queries rewritten follow the consistent style (e.g., always use '!=' instead of '!=' and '<>' interchangeably), the Codex performance may still be improved given rewritten queries in the prompt.

Dataset	Template	Random	SrcSelfDis
GeoQuery	Question	37	43
	SQL	45	50
Advising	Question	39	49
	SQL	50	50
ATIS	Question	27	30
	SQL	43	50

Table 4: The number of question and query templates in the 50 examples selected randomly and by SrcSelfDis. With SrcSelfDis selection, more question and SQL templates are selected.

plates are selected when the selection method SrcSelfDis is employed. It indicates the effectiveness of SrcSelfDis in improving the diversity of the selected examples. Similar results are also observed when K-means, SelfDis and Agglo selection methods are used.

C Experiment Details

C.1 Computing Infrastructure

All experiments are implemented on RTX 3090 GPUs with 32GB memory.

C.2 Models

We use the pre-trained T5-large model from Huggingface[6] and the version trained on Spider from Scholak et al. (2021)[7] to calculate input embeddings and finetune downstream tasks. Both versions of T5-large models contain 770 million parameters. For the in-context learning, we use Codex[8] with 12 billion parameters.

C.3 Hyperparameters

We report experiment hyperparameters in Table 5. Since in the low-resource regime in terms of annotation budgets and computational resources, we consider it more worthwhile to use all available labeled instances for training T5 or construct prompts for Codex without validation split. Therefore, we use the same set of hyperparameters in all experiments.

C.4 Computational Budgets

We report the time to finetune and evaluate T5-large model, and evaluate Codex model in nine datasets.

[6]https://huggingface.co/t5-large
[7]https://huggingface.co/tscholak/1wnr382e
[8]https://openai.com/api/

Hyperparameter	**Assignment**
Fituning (T5-large)	
epochs	30
learning rate	1e-4
per device train batch size	2
beam size	1
maximum generation length	512
gradient accumulation steps	2
In-context Learning (Codex)	
temparature	1
maximum generation tokens	500
top probability	0.95
frequency penalty	0.0
presence penalty	0.0
stop token	"#",";"

Table 5: Hyperarameters used in finetuning and in-context learning across nine datasets.

D Exact Experimental Results

We report the exact numbers of Fig 2, Fig 3, Fig 4, Fig 5, and Fig 6 in Table 7, Table 8, Table 9, Table 10, and Table 11 respectively.

E Dataset Examples

E.1 Spider

Question:

Find the first and last names of the students who are living in the dorms that have a TV Lounge as an amenity.

SQL query:

```
SELECT T1.fname, T1.lname FROM
student AS T1 JOIN lives_in AS T2
 ON T1.stuid = T2.stuid WHERE T2.
dormid IN (SELECT T3.dormid FROM
has_amenity AS T3 JOIN
dorm_amenity AS T4 ON T3.amenid =
 T4.amenid WHERE T4.amenity_name
= 'TV Lounge')
```

E.2 GeoQuery

Question:

State the state with the largest area

SQL query:

```
SELECT STATE_NAME FROM STATE
WHERE AREA = ( SELECT AREA FROM
STATE ORDER BY AREA DESC LIMIT 1)
```

E.3 Advising

Question:

	GeoQuery	Advising	ATIS	Scholar	IMDB	Kaggle	Restaurants	Yelp	Academic
Finetuning (T5-spider)									
Finetune	309	312	312	308	311	309	307	310	309
Evaluation	214	428	3416	248	38	116	114	51	70
In-context Learning (Codex)									
Evaluation	1236	1568	7580	1164	213	792	724	296	85

Table 6: Experiment time in the unit of second.

method	number	GeoQuery	Advising	ATIS
Random	50	$58.8_{3.8}$	$16.6_{6.8}$	$34.2_{5.0}$
SrcSelfDis	50	68.0	26.4	40.6
Random	250	$77.3_{2.2}$	$59.6_{2.4}$	$38.2_{2.2}$
SrcSelfDis	250	80.2	69.0	48.6
Random	500	$81.1_{1.4}$	$81.3_{1.3}$	$42.4_{1.9}$
SrcSelfDis	500	83.4	85.8	53.2

Table 7: Impact of selection numbers on fine-tuning.

method	number	GeoQuery	Advising	ATIS
Random	50	$57.9_{2.1}$	$16.2_{3.2}$	$31.6_{2.3}$
SelfDis	50	$65.4_{1.1}$	$20.6_{2.6}$	$36.7_{2.2}$
Random	250	$69.4_{1.0}$	$56.7_{2.5}$	$37.5_{2.2}$
SelfDis	250	$75.4_{0.9}$	$59.2_{2.1}$	$41.7_{1.6}$
Random	500	$80.4_{1.0}$	$75.2_{1.4}$	$40.2_{1.7}$
SelfDis	500	$81.2_{0.8}$	$79.2_{1.0}$	$43.8_{1.1}$

Table 8: Impact of selection numbers on in-context learning.

Embedding	GeoQuery	Advising	ATIS
Finetuning (T5)			
T5-trained	68.0	26.4	40.6
T5	63.1	20.5	36.6
SimCSE	61.1	19.3	35.3
In-context Learning (Codex)			
T5-trained	$60.0_{1.8}$	$18.3_{2.6}$	$36.7_{2.2}$
T5	$58.8_{1.9}$	$17.3_{2.8}$	$32.5_{2.3}$
SimCSE	$58.3_{2.0}$	$16.6_{2.9}$	$31.8_{2.3}$

Table 9: Impact of embedding methods.

Similarity	GeoQuery	Advising	ATIS
Finetuning (T5)			
Consine	68.0	26.4	40.6
Euclidean	41.3	7.5	16.6
Dot	45.1	8.4	17.5
In-context Learning (Codex)			
Consine	$60.0_{1.8}$	$18.3_{2.6}$	$36.7_{2.2}$
Euclidean	$22.3_{5.6}$	$0.8_{0.4}$	$12.6_{2.1}$
Dot	$24.4_{4.4}$	$1.1_{0.3}$	$14.7_{1.8}$

Table 10: Impact of similarity measures.

Model	Query	GeoQuery	Advising	ATIS
T5	rewritten	68.0	26.4	40.6
T5	not rewritten	60.5	18.2	35.8
Codex	rewritten	$60.0_{1.8}$	$18.3_{2.6}$	$36.7_{2.2}$
Codex	not rewritten	$59.8_{1.6}$	$17.6_{2.3}$	$35.5_{1.4}$

Table 11: Impact of query style on model performance.

Which is the easiest class to get my Core requirement fulfilled

SQL query:

```
SELECT DISTINCT T2.DEPARTMENT ,
T2.NAME , T2.NUMBER , T1.WORKLOAD
 , T1.WORKLOAD FROM
PROGRAM_COURSE AS T1 JOIN COURSE
AS T2 ON T1.COURSE_ID = T2.
COURSE_ID WHERE T1.CATEGORY LIKE
"Core" AND T1.WORKLOAD = ( SELECT
 WORKLOAD FROM PROGRAM_COURSE
WHERE CATEGORY LIKE "Core" ORDER
BY WORKLOAD LIMIT 1)
```

E.4 ATIS

Question:

Are there any flights on 6 10 from BURBANK to TACOMA

SQL query:

```
SELECT DISTINCT T3.FLIGHT_ID FROM
 CITY AS T1 JOIN AIRPORT_SERVICE
AS T2 ON T1.CITY_CODE = T2.
CITY_CODE JOIN FLIGHT AS T3 ON T3.
TO_AIRPORT = T2.AIRPORT_CODE JOIN
 AIRPORT_SERVICE AS T4 ON T3.
FROM_AIRPORT = T4.AIRPORT_CODE
JOIN DAYS AS T5 ON T3.FLIGHT_DAYS
 = T5.DAYS_CODE JOIN CITY AS T6
ON T6.CITY_CODE = T4.CITY_CODE
JOIN DATE_DAY AS T7 ON T5.
DAY_NAME = T7.DAY_NAME WHERE T1.
CITY_NAME = "TACOMA" AND T7.
DAY_NUMBER = 10 AND T7.
MONTH_NUMBER = 6 AND T7.YEAR =
1991 AND T6.CITY_NAME = "BURBANK"
```

E.5 Academic

Question:

Return me the authors who have more than 10 papers in PVLDB

SQL query:

```
SELECT T4.NAME FROM PUBLICATION
AS T1 JOIN JOURNAL AS T2 ON T1.
JID = T2.JID JOIN WRITES AS T3 ON
 T3.PID = T1.PID JOIN AUTHOR AS
T4 ON T3.AID = T4.AID WHERE T2.
NAME = \"PVLDB\" GROUP BY T4.NAME
 HAVING COUNT( DISTINCT ( T1.
TITLE ) ) > 10
```

E.6 Yelp

Question:

List all the businesses with more than 4.5 stars

SQL query:

```
SELECT NAME FROM BUSINESS WHERE
RATING > 4.5
```

E.7 IMDB

Question:

Find all movies directed by “Steven Spielberg" after 2006

SQL query:

```
SELECT T3.TITLE FROM DIRECTOR AS
T1 JOIN DIRECTED_BY AS T2 ON T1.
DID = T2.DID JOIN MOVIE AS T3 ON
T3.MID = T2.MSID WHERE T1.NAME =
"Steven Spielberg" AND T3.
RELEASE_YEAR > 2006
```

E.8 Scholar

Question:

What papers have been written by Peter Mertens and Dina Barbian

SQL query:

```
SELECT DISTINCT T1.PAPERID FROM
WRITES AS T1 JOIN AUTHOR AS T2 ON
 T1.AUTHORID = T2.AUTHORID JOIN
WRITES AS T3 ON T3.PAPERID = T1.
PAPERID JOIN AUTHOR AS T4 ON T3.
AUTHORID = T4.AUTHORID WHERE T2.
AUTHORNAME = "Peter Mertens" AND
T4.AUTHORNAME = "Dina Barbian"
```

E.9 Restaurants

Question:

Where is the best restaurant in san francisco for french food

SQL query:

```
SELECT T2.HOUSE_NUMBER , T1.NAME
FROM RESTAURANT AS T1 JOIN
LOCATION AS T2 ON T1.ID = T2.
RESTAURANT_ID WHERE T2.CITY_NAME
= "san francisco" AND T1.
FOOD_TYPE = "french" AND T1.
RATING = ( SELECT T1.RATING FROM
RESTAURANT AS T1 JOIN LOCATION AS
 T2 ON T1.ID = T2.RESTAURANT_ID
WHERE T2.CITY_NAME = "san
francisco" AND T1.FOOD_TYPE = "
french" ORDER BY T1.RATING DESC
LIMIT 1)
```

E.10 Kaggle

Question:

Which states have produced the largest number of candidates inducted into the hall of fame

SQL query:

```
SELECT T2.birth_state FROM player
 AS T2 JOIN hall_of_fame as T1 ON
 T1.player_id = T2.player_id
WHERE inducted = "Y" GROUP BY T2.
birth_state ORDER BY count(T1.
player_id) DESC LIMIT 1
```