Skill-Based Few-Shot Selection for In-Context Learning

Shengnan An[*♢,♣], **Bo Zhou**[*♡,♣], **Zeqi Lin**[†♣], **Qiang Fu**[♣], **Bei Chen**[♣], **Nanning Zheng**[†♢], **Weizhu Chen**[♣], **Jian-Guang LOU**[♣]

[♢]National Key Laboratory of Human-Machine Hybrid Augmented Intelligence, National Engineering Research Center of Visual Information and Applications, Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University

[♣]Microsoft Corporation, [♡]Northeastern University

[♢]{an1006634493@stu, nnzheng@mail}.xjtu.edu.cn, [♡]2171386@stu.neu.edu.cn

[♣]{Zeqi.Lin, qifu, beichen, wzchen, jlou}@microsoft.com

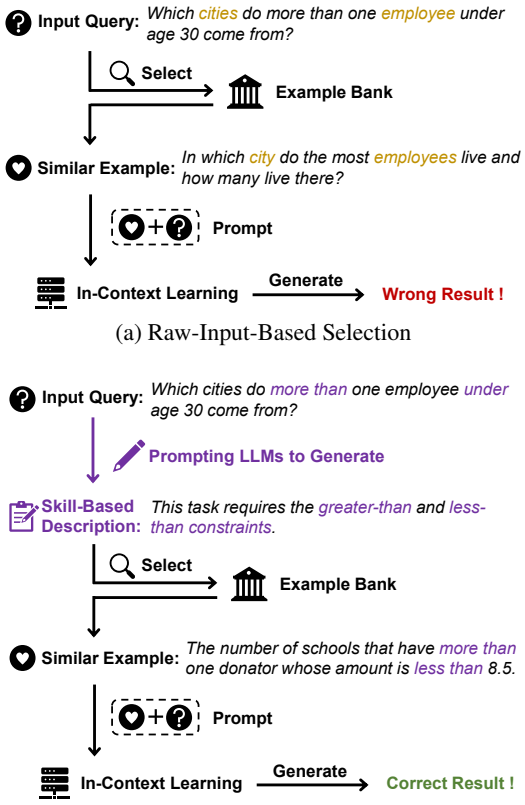
Abstract

In-context learning is the paradigm that adapts large language models to downstream tasks by providing a few examples. *Few-shot selection*—selecting appropriate examples for each test instance separately—is important for in-context learning. In this paper, we propose SKILL-KNN, a skill-based few-shot selection method for in-context learning. The key advantages of SKILL-KNN include: (1) it addresses the problem that existing methods based on pre-trained embeddings can be easily biased by surface natural language features that are not important for the target task; (2) it does not require training or fine-tuning of any models, making it suitable for frequently expanding or changing example banks. The key insight is to optimize the inputs fed into the embedding model, rather than tuning the model itself. Technically, SKILL-KNN generates the skill-based descriptions for each test case and candidate example by utilizing a pre-processing few-shot prompting, thus eliminating unimportant surface features. Experimental results across five cross-domain semantic parsing datasets and six backbone models show that SKILL-KNN significantly outperforms existing methods.

1 Introduction

In-context learning (Brown et al., 2020) has become a prevailing paradigm for utilizing large language models (LLMs) (Hendrycks et al., 2020; Patel and Pavlick, 2021; Rae et al., 2021; Zhang et al., 2022a; Hoffmann et al., 2022; Srivastava et al., 2022; Chowdhery et al., 2022; Smith et al., 2022; Wei et al., 2022a). It employs a frozen task-agnostic backbone model to serve various downstream tasks without requiring parameter updates for each task. Under in-context learning, the LLMs generate output for an input query by conditioning on the prompt that contains input-output examples. Due to limited context length of the language model, only a few examples can be presented in the prompt. Prior studies have found that the performance of in-context learning is sensitive to the selected in-context examples (Liu et al., 2022; Zhang et al., 2022b; Chen et al., 2023b). Therefore, one essential research question is: *how to select proper examples from a large example bank?*

Raw-input-based selection is one widely applied solution (Gao et al., 2021; Liu et al., 2022; Hu et al., 2022). It involves embedding raw inputs of examples using an off-the-shelf embedding model



(a) Raw-Input-Based Selection

(b) Skill-Based Selection

Figure 1: In-context learning with different selection methods. (a) Examples from raw-input-based selection just share similar entities with the input query. (b) With the skill-based description, the selected examples contain the desired task-specific skills.

[*] Work done during the internship at Microsoft.

[†] Corresponding authors.

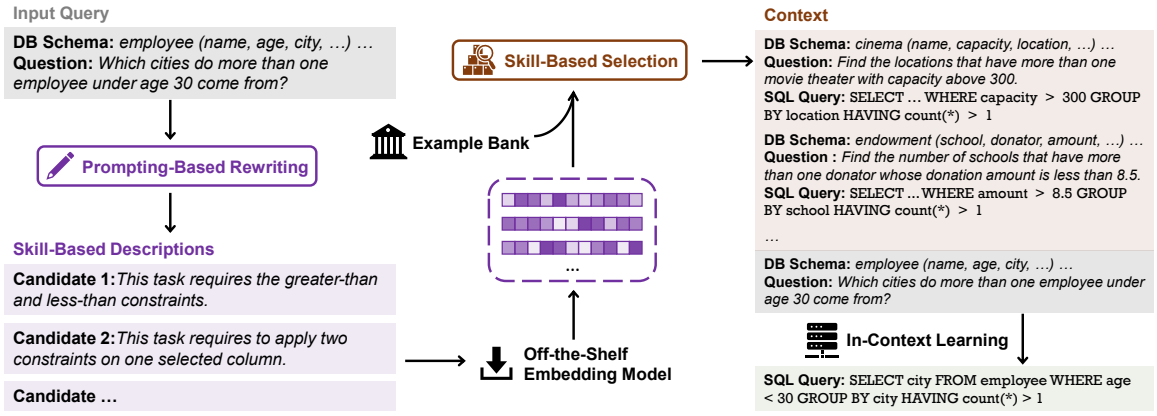


Figure 2: The bird's-eye view of SKILL-KNN, a rewrite-then-retrieve selection method to facilitate in-context learning with skill-based descriptions.

and then selecting the most similar examples. It can be conveniently applied in various downstream tasks. However, this method can be easily biased by surface natural language features that are not important for the target task. For instance, in semantic parsing tasks[1], the raw-input-based selection just finds out examples with similar entities (as illustrated in Figure 1a), while the better in-context examples should contain the required executable operations in logical forms, which can be regarded as the *task-specific skills*.

To overcome this limitation, we aim to make this embedding-based selection better aware of the intrinsic skills behind the raw inputs. We consider to harness the power of **prompting LLMs to convert the desired skills from raw inputs**, which maintains the training-free advantage during selection. There has been much work trying to fine-tune the embedding model for each task based on the example bank (Rubin et al., 2022; Poesia et al., 2022; Hu et al., 2022; Ye et al., 2023). However, fine-tuning-based methods are difficult to apply in practical scenarios: it is laborious to train and save the embedding model for each task, and it is also inconvenient to re-train the model on a dynamic example bank that can be updated frequently.

Specifically, we introduce **SKILL-KNN**, a training-free, skill-based selection method (briefly illustrated in Figure 1b). Overall, SKILL-KNN will first **generate skill-based descriptions from raw input queries**, then feed these descriptions into an off-the-shelf embedding model to select most similar examples. To generate skill-based descriptions, we prompt a frozen LLM with just a few human-annotated demonstrations, which does not require any fine-tuning process and has no rule-based constraint. Additionally, to alleviate the sensitivity to the order of annotated demonstrations during generation, we design two variants of SKILL-KNN: we sample a set of candidate descriptions by shuffling annotated demonstrations, then select candidate based on **consistency** and **distinctiveness**, respectively.

The experimental results show that SKILL-KNN brings a considerable boost for in-context learning compared to the raw-input-based selection. We evaluate SKILL-KNN on five challenging semantic parsing datasets: Spider (Yu et al., 2018b), Dr. Spider (Chang et al., 2023), KaggleDBQA (Lee et al., 2021), BIRD (Li et al., 2023c), and COGS (Kim and Linzen, 2020). We take six models for in-context learning: text-chat-davinci-002, code-davinci-002, text-davinci-003, code-cushman-002, gpt-35-turbo, and gpt-4. Across these tasks and models, SKILL-KNN consistently performs best among non-oracle selection methods and, at times, is even comparable to oracle methods. For instance, with text-chat-davinci-002, SKILL-KNN achieves 78.3% execution accuracy on Spider, while the best raw-input-based selection method reaches 74.6% and Target-KNN[2] attains 78.6%. Furthermore, our ablation study indicates that SKILL-KNN retains its superiority when constraints are imposed on the annotated demonstrations, including reducing the number of demonstrations, restricting the database diversity, and decreasing the operation coverage.

Our contributions are three-fold: 1) we propose a skill-based few-shot selection method SKILL-KNN, which leverages the power of prompting

[1]Semantic parsing means to parse an NL utterance into a machine-understandable logical form (e.g., a SQL query).

[2]One of the oracle methods, detailed in Section 4.2.

Table 1: Part of our annotated skill-based descriptions for text-to-SQL tasks.

Input Query	Database Schema	Skill-Based Description
Show all majors.	allergy type [allergy, allergytype] has allergy [stuid, allergy] student [stuid, lname, fname, age, sex, major, ...]	To solve this task in the database, we need to select distinct values in the column.
Count the number of different colleges that players who play for Columbus Crew are from.	team [team id, name] country [country id, country name, capital, ...] match season [season, player, position, country, team, ...] ...	To solve this task in the database, we need to join two tables and count the number of distinct values in the column.
Which catalog contents have a product stock number that starts from "2"? Show the catalog entry names.	catalogs [catalog id, catalog name, catalog publisher, ...] catalog structure [catalog level number, catalog id, ...] catalog contents [catalog entry id, catalog level number, ...] ...	To solve this task in the database, we need to select one column and apply a constraint on the format of values in this column.

LLMs to generate skill-based descriptions; 2) we design two variants of SKILL-KNN based on consistency and distinctiveness, respectively; 3) our comprehensive experiments across various semantic parsing tasks and backbone models demonstrate the effectiveness of SKILL-KNN, and our analysis of annotated demonstrations provides further insights for better utilization of SKILL-KNN.

2 Preliminaries

In this section, we introduce embedding-based few-shot selection as the preliminary of our method.

2.1 In-Context Learning with Few-Shot Selection

Consider a downstream task $\mathcal{T}$ that contains a set of input-output examples $\{(x_i \rightarrow y_i)\}^n$ (termed *example bank* $\mathcal{B}$)[3] , and a pre-trained large language model with frozen parameters θ. Given a test input query x_t, the large language model with in-context learning will generate an output y_t by sampling from the following distribution,

$$y_t \sim \text{LLM}_{\theta,\tau}[R(x_t, \mathcal{B}) \oplus x_t], \quad (1)$$

in which τ is the sampling temperature, $R(x_t, \mathcal{B})$ returns a sequence of examples selected from $\mathcal{B}$ according to x_t, and $\oplus$ means to sequentially concatenate two sequences. In the later part of the paper, we omit the frozen θ and set $\tau = 0$ by default, which means to perform greedy decoding.

Few-shot selection aims to design the algorithm $R(x_t, \mathcal{B})$ that can work well for task $\mathcal{T}$.

2.2 Embedding-Based Few-Shot Selection

A standard implementation of $R(x_t, \mathcal{B})$ is to leverage an off-the-shelf embedding model $\text{Emb}(\cdot)$ and calculate the embedding similarity of raw inputs (Liu et al., 2022),

$$\text{sim}(x_t, x_i) = \frac{\text{Emb}(x_t)\text{Emb}(x_i)^T}{|\text{Emb}(x_t)||\text{Emb}(x_i)|}, \quad (2)$$

in which x_i is the input[4] of one example $(x_i, y_i) \in \mathcal{B}$. Based on Equation 2, we can select k most similar examples from $\mathcal{B}$. In addition, these examples will be sorted in the prompt according to their similarities to the test input query: examples with higher similarity scores will be placed closer to the test input query.

This standard implementation of $R(x_t, \mathcal{B})$ is a raw-input-based selection. It just searches for examples with similar inputs (i.e., the x_t and x_i in Equation 2). Some recent researches propose to fine-tune the embedding model (from $\text{Emb}(\cdot)$ to $\text{Emb}'(\cdot)$) (Rubin et al., 2022; Poesia et al., 2022; Hu et al., 2022; Ye et al., 2023). In this paper, we want to explore how to improve the effectiveness of few-shot selection without training or fine-tuning of any models.

3 SKILL-KNN

SKILL-KNN involves a rewrite-then-retrieve process to better exploit the potential of in-context learning. Figure 2 gives a bird's-eye view of our method. To mine and utilize task-specific skills, SKILL-KNN contains a **prompting-based rewriting** stage and a **skill-based selection** stage. Prompting-based rewriting will prompt LLMs to generate skill-based descriptions from the given input query. Skill-based selection will return few-shot examples based on these generated descriptions. In the following, we elaborate the design of SKILL-KNN.

3.1 Generating Skill-Based Descriptions

We prompt a frozen large language model to rewrite each input query as a skill-based description, which does not require any fine-tuning process. Specifically, we first annotate the skill-based descriptions for 16 examples in $\mathcal{B}$, then prompt the large lan-

[3] In semantic parsing tasks, each input query contains a natural language question along with the database schema.

[4] For raw-input-based selection, we only use the question in the input query for embedding and omit the database schema, as it contains much trivial and redundant information for the question and could confuse the embedding model.

guage model with these annotated demonstrations and generate for other examples in $\mathcal{B}$ and for each test input query.

Note that we annotated skills with natural language descriptions rather than rule-based constraints. It is important to note that off-the-shelf embedding models are primarily pre-trained on natural language (NL) data and may not be well-suited for handling specifically designed structural constraints. By annotating skills with NL descriptions, we can better align with the off-the-shelf embedding models, which in turn allows us to leverage their generalizability when encoding unannotated NL descriptions more effectively. Thus, these natural language skills can better suit the off-the-shelf embedding model, and our annotated demonstrations can better generalize to unseen data.

Formally, with a set of annotated demonstrations $\{x_a \rightarrow s_a\}$ in which s_a is the annotated skill-based description for the raw input x_a, we generate the s_i for each unannotated input x_i by prompting the large language model,

$$s_i = \text{LLM}[\{x_a \rightarrow s_a\} \oplus x_i], \tag{3}$$

then, these descriptions are fed into the off-the-shelf embedding model to select similar examples,

$$\text{sim}(s_t, s_i) = \frac{\text{Emb}(s_t)\text{Emb}(s_i)^T}{|\text{Emb}(s_t)||\text{Emb}(s_i)|}. \tag{4}$$

Table 1 shows part of our annotated demonstrations for text-to-SQL tasks, and all our annotations are contained in Appendix E. Note that different text-to-SQL tasks can share the same set of annotated demonstrations in our experiments.

Equation 4 defines the basic version of SKILL-KNN. Moreover, we notice that the generated skill-based descriptions sometimes could be sensitive to the order of annotated demonstrations. Such a sensitivity is also observed in some previous work (Zhao et al., 2021; Lu et al., 2022). Therefore, we design two variants of SKILL-KNN to further address this sensitivity issue.

3.2 Variants

To alleviate the influence from the sensitivity to prompt order, we design two variants of SKILL-KNN that change the order of annotated demonstrations and perform rewriting multiple times. Specifically, for each input x_i, both two variants generate a set of candidate descriptions $S_i = \{s_i^1, s_i^2, ..., s_i^m\}$ according to Equation 3 by changing the order in $\{x_a \rightarrow s_a\}$. Then, two variants use these candidate descriptions from the view of consistency and distinctiveness, respectively. Figure 3 briefly illustrates the basic ideas behind these two variants. Appendix B.1 provides further analysis for the motivation behind the design of two variants.

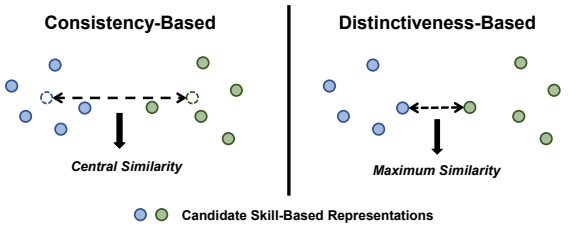


Figure 3: Two variants of SKILL-KNN. The blue and green points represent two candidate sets of skill-based representations.

Consistency-Based Variant. From the view of consistency, we take the central embedding of all candidate descriptions during selecting examples,

$$\text{sim}_c(S_t, S_i) = \frac{\overline{e_t}\,\overline{e_i}^T}{|\overline{e_t}|\,|\overline{e_i}|}, \overline{e} = \frac{1}{m}\sum_{s^j \in S} \text{Emb}(s^j), \tag{5}$$

in which S_t and S_i represent two sets of candidate descriptions for the test input x_t and one example $(x_i, y_i) \in \mathcal{B}$, respectively. This variant is inspired by prior work on improving the consistency of chain-of-thought reasoning (Wang et al., 2022; Li et al., 2022a). As illustrated in the left in Figure 3, Equation 5 can be regarded as an embedding-level majority voting among all candidate descriptions during selection.

Distinctiveness-Based Variant. Considering that the central embedding can sometimes be overwhelmed by trivial candidates, we want to highlight the most distinctive and informative description among all candidates. Formally, we consider the maximum similarity score between two sets for selection,

$$\text{sim}_d(S_t, S_i) = \max_{j,k} \frac{\text{Emb}(s_t^j)\text{Emb}(s_i^k)^T}{|\text{Emb}(s_t^j)||\text{Emb}(s_i^k)|}, \tag{6}$$

in which $s_t^j \in S_t$ and $s_i^k \in S_i$. As illustrated in the right in Figure 3, Equation 6 means that we take the minimum distance among two set of candidates for selecting similar examples.

4 Experimental Setup

In this section, we will introduce the tasks, compared selection methods, backbone models, and

Table 2: Our main experimental results (%) across various LLMs and tasks. Numbers in **bold** are the best results across non-oracle methods, and results with underlines can outperform at least one oracle method.

Backbone	Method	Spider	Dr. Spider			KDBQA	BIRD	COGS		*#Wins*
			DB	NLQ	SQL			P.S.	P.A.	
text-chat-davinci-002	Random	72.9	54.1	58.1	68.2	24.1	35.7	59.1	61.5	0
	KNN w/ SBERT (Liu et al., 2022)	73.0	51.6	58.2	67.3	24.3	38.3	78.5	71.1	0
	KNN w/ OpenAI Babbage (Liu et al., 2022)	74.0	53.2	61.0	69.2	35.3	38.1	88.3	74.1	0
	KNN w/ OpenAI Ada (Liu et al., 2022)	72.8	51.8	59.2	69.6	31.9	37.1	81.3	64.2	0
	MMR w/ OpenAI Ada (Ye et al., 2022)	74.6	54.1	60.7	71.3	37.3	37.2	86.8	78.9	0
	SKILL-KNN w/ SBERT (base)	76.8	55.3	60.3	72.1	34.7	38.9	93.8	88.3	0
	+ consistency	76.0	54.8	60.3	71.9	33.8	38.2	94.3	87.9	0
	+ distinctiveness	**78.3**	**57.0**	**61.4**	**72.2**	**40.0**	38.0	92.6	88.8	5
	SKILL-KNN w/ OpenAI Ada (base)	77.1	55.4	60.5	70.9	38.7	**38.9**	94.9	95.6	1
	+ consistency	77.2	54.8	59.6	71.1	**40.0**	38.3	**95.2**	93.5	2
	+ distinctiveness	77.2	56.5	59.8	70.1	39.5	38.3	93.3	**97.0**	1
	Target-KNN (oracle)	78.6	56.6	65.5	75.5	37.8	41.7	86.8	83.2	-
	Target Sketch Matching (oracle)	79.5	54.2	65.1	76.2	40.4	40.8	96.9	95.7	-
code-davinci-002	Random	74.2	53.9	59.3	70.2	27.9	38.4	56.1	63.1	0
	KNN w/ SBERT (Liu et al., 2022)	73.1	51.2	58.6	69.1	31.2	39.9	75.4	64.7	0
	KNN w/ OpenAI Babbage (Liu et al., 2022)	73.6	50.7	59.6	69.1	37.5	38.2	85.4	72.4	0
	KNN w/ OpenAI Ada (Liu et al., 2022)	72.7	50.4	60.2	69.5	35.7	38.2	77.3	60.3	0
	MMR w/ OpenAI Ada (Ye et al., 2022)	74.8	53.7	60.7	69.4	37.8	37.9	84.9	75.4	0
	SKILL-KNN w/ SBERT (base)	76.4	53.0	60.2	**72.4**	38.6	40.2	93.1	86.8	1
	+ consistency	77.1	51.1	60.7	**72.4**	36.8	40.1	93.5	87.1	1
	+ distinctiveness	**77.4**	**55.0**	60.9	71.0	**43.0**	38.9	91.9	88.4	3
	SKILL-KNN w/ OpenAI Ada (base)	76.2	54.7	**61.4**	70.2	39.8	40.3	94.0	92.9	1
	+ consistency	76.4	54.9	60.2	71.8	39.5	**40.8**	**96.2**	88.8	2
	+ distinctiveness	76.6	**55.0**	60.6	70.2	38.9	**40.8**	93.5	**94.0**	3
	Target-KNN (oracle)	78.8	56.1	68.2	76.4	44.5	44.3	85.9	79.7	-
	Target Sketch Matching (oracle)	80.9	54.0	66.7	77.9	44.9	43.8	95.0	94.8	-
text-davinci-003	Random	69.0	52.2	55.1	64.5	20.6	36.0	65.5	61.2	0
	KNN w/ SBERT (Liu et al., 2022)	69.9	50.2	56.1	67.2	21.0	37.7	82.1	71.1	0
	KNN w/ OpenAI Babbage (Liu et al., 2022)	72.2	53.4	58.0	69.5	26.8	37.7	88.8	77.2	0
	KNN w/ OpenAI Ada (Liu et al., 2022)	70.8	50.3	57.2	66.3	30.3	36.5	82.8	64.2	0
	MMR w/ OpenAI Ada (Ye et al., 2022)	72.4	53.3	58.6	69.9	31.4	37.7	87.1	81.0	0
	SKILL-KNN w/ SBERT (base)	74.9	54.4	59.0	70.2	32.9	38.0	94.8	88.4	0
	+ consistency	75.3	54.8	59.0	70.6	32.0	38.1	94.3	89.2	0
	+ distinctiveness	**76.6**	54.1	58.9	70.6	**36.8**	37.4	93.3	87.5	2
	SKILL-KNN w/ OpenAI Ada (base)	74.2	55.2	**59.5**	68.5	34.0	38.4	95.7	94.3	1
	+ consistency	74.3	55.0	**59.5**	**71.0**	**36.8**	**40.2**	**96.7**	92.7	5
	+ distinctiveness	73.7	**56.2**	59.4	70.6	32.4	37.9	93.8	**97.4**	2
	Target-KNN (oracle)	75.6	54.2	63.9	73.3	35.1	40.0	87.8	84.5	-
	Target Sketch Matching (oracle)	76.4	52.5	61.4	72.0	31.6	39.8	97.6	95.3	-
code-cushman-002	Random	72.2	51.5	56.6	66.8	26.1	35.3	56.7	55.6	0
	KNN w/ SBERT (Liu et al., 2022)	67.6	47.8	54.4	64.6	29.4	37.0	70.3	62.5	0
	KNN w/ OpenAI Babbage (Liu et al., 2022)	71.6	48.6	56.4	66.8	36.4	36.4	77.5	71.6	0
	KNN w/ OpenAI Ada (Liu et al., 2022)	68.9	48.1	57.3	67.4	31.9	34.8	71.3	54.7	0
	MMR w/ OpenAI Ada (Ye et al., 2022)	69.3	49.8	57.1	68.6	36.2	36.8	75.6	72.8	0
	SKILL-KNN w/ SBERT (base)	74.5	52.1	58.0	69.3	35.1	37.0	90.6	84.1	0
	+ consistency	**74.7**	52.3	58.0	69.6	35.3	38.3	91.1	86.6	1
	+ distinctiveness	72.8	52.0	**58.8**	67.2	**40.8**	35.9	91.8	83.2	2
	SKILL-KNN w/ OpenAI Ada (base)	73.6	52.4	58.4	69.0	38.4	37.9	93.2	86.6	0
	+ consistency	73.5	**53.0**	57.9	**69.8**	40.0	**38.6**	**95.0**	82.8	4
	+ distinctiveness	73.7	51.2	57.6	67.9	38.9	37.3	91.9	**90.9**	1
	Target-KNN (oracle)	77.6	52.5	65.6	73.8	40.8	41.7	77.0	72.8	-
	Target Sketch Matching (oracle)	77.2	51.7	62.2	73.2	39.7	39.8	91.9	94.0	-

hyper-parameters in our experiments.

4.1 Tasks

We evaluate on five challenging cross-domain semantic parsing datasets. Due to the cross-domain property, the model can not easily solve these tasks by just copying some similar surface features from the provided in-context examples.

Spider (Yu et al., 2018b) is a large-scale text-to-SQL dataset. It contains a train set with 7,000 examples and a dev set with 1,034 examples. Moreover, the train set and dev set do not share any database. We take the train set of Spider as the example bank, and evaluate on the dev set.

Dr. Spider (Chang et al., 2023) is a diagnostic evaluation benchmark constructed based on Spider. It contains 15,269 examples which can be divided into 3 sub-tasks, according to the type of designed perturbations: database perturbations (DB), natural language question perturbations (NLQ), and SQL query perturbations (SQL). We take the train set of Spider as the example bank, since Dr. Spider is purely an evaluation benchmark.

KaggleDBQA (Lee et al., 2021) (KDBQA) is a small while complex dataset towards realistic evaluation of text-to-SQL semantic parsers. It contains 8 real Web databases with original formatting and

Table 3: Performance of gpt-35-turbo and gpt-4 on Spider dev set.

Backbone	Method	Exec. Acc.
gpt-4	Random	76.1
	KNN w/ SBERT (Liu et al., 2022)	76.7
	Din-SQL (Pourreza and Rafiei, 2023)	74.2
	SKILL-KNN w/ SBERT (base)	81.3
	+ consistency	**82.7**
	+ distinctiveness	82.3
gpt-35-turbo	Random	74.3
	KNN w/ SBERT (Liu et al., 2022)	73.7
	KNN w/ OpenAI Babbage (Liu et al., 2022)	73.4
	SKILL-KNN w/ SBERT (base)	76.2
	+ consistency	76.3
	+ distinctiveness	**76.8**

Table 4: Comparison with the fine-tuning-based selection methods on Spider dev set.

Backbone	Method	Exec. Acc.
text-chat-davinci-002	EPR (Rubin et al., 2022)	74.4
	CEIL (Ye et al., 2023)	75.0
	TST (Poesia et al., 2022)	76.3
	SKILL-KNN w/ SBERT (base)	76.8
code-davinci-002	EPR (Rubin et al., 2022)	75.9
	SKILL-KNN w/ SBERT (base)	76.4
	+ consistency	77.1
	+ distinctiveness	**77.4**
text-davinci-003	EPR (Rubin et al., 2022)	69.7
	SKILL-KNN w/ SBERT (base)	74.9
	+ consistency	75.3
	+ distinctiveness	**76.6**
code-cushman-002	EPR (Rubin et al., 2022)	74.6
	SKILL-KNN w/ SBERT (base)	74.5
	+ consistency	**74.7**
	+ distinctiveness	72.8

275 unrestricted questions. Since ther is not much data in KDBQA, we take it as a pure test set and use the train set of Spider as the example bank.

BIRD (Li et al., 2023c) is a large scale text-to-SQL dataset with real-world database contents. It has 9,428 training examples and 1,534 test cases in the dev set. Each case in BIRD is equipped with a description of the required external knowledge, which is not contained in above three text-to-SQL tasks. Since the database schema in BIRD is too large, we first take grounding to reduce the size of schema (detailed in Appendix D.4).

COGS (Kim and Linzen, 2020) is a synthetic benchmark for testing compositional generalization in semantic parsing. It can also be regarded as a cross-domain setting, containing a significant distribution shift between train set and test set. The logical form in COGS represents the thematic roles in

Table 5: Performance of SKILL-KNN and two baselines on GSM8K.

Backbone	Method	Accuracy
text-chat-davinci-002	Random	69.1
	KNN w/ SBERT (Liu et al., 2022)	69.9
	SKILL-KNN w/ SBERT (base)	71.0

the input query (detailed in Appendix D.3). We use the output format designed in An et al. (2023) and evaluate on two sub-tasks, primitive substitution (P.S.) and primitive structural alternation (P.A.).

4.2 Selection Methods

We mainly compare SKILL-KNN with training-free selection methods.

Random. We randomly select examples from $\mathcal{B}$ as in-context examples. For each test case, we take random selections 3 times and average the results.

KNN (Liu et al., 2022). We test three off-the-shelf embedding models: Sentence-BERT (SBERT) with all-mpnet-base-v2 checkpoint[5] (Reimers and Gurevych, 2019), OpenAI embedding model[6] with text-similarity-babbage-001 checkpoint (OpenAI Babbage), and OpenAI embedding model with text-embedding-ada-002 checkpoint (OpenAI Ada). KNN with OpenAI embedding models can serve as strong baselines for training-free selection methods, as these large models have been well pre-trained for judging text similarity (Neelakantan et al., 2022).

MMR (Ye et al., 2022) is a dynamic selection method to enhance the diversity of selected examples from KNN. It adds a penalty term according to the similarity to the already selected examples. We take OpenAI Ada for embedding and follow the implementation details in Ye et al. (2022).

SKILL-KNN (ours). We test SKILL-KNN with SBERT and OpenAI Ada. For the base version of SKILL-KNN (i.e., without consistency or distinctiveness), we shuffle the order of annotated demonstrations to generate $m = 5$ skill-based descriptions for each input query and average the results. There is a balance between achieving optimal performance and minimizing computational costs. We provide more experimental analysis in Appendix B.2. For two variants, we take all 5 generated descriptions as the candidate set.

[5] https://www.sbert.net/.

[6] https://platform.openai.com/docs/guides/embeddings/what-are-embeddings.

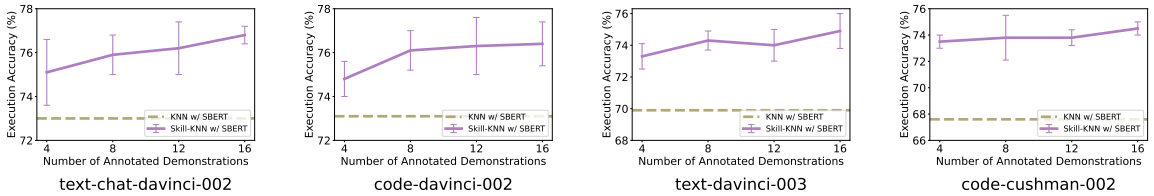


Figure 4: Performance of SKILL-KNN (base version) with different number of annotated demonstrations.

Besides these training-free baselines, we also compare with three fine-tuning-based methods. These methods leverage the example bank for fine-tuning the embedding model.

EPR (Rubin et al., 2022) requires a scoring LM to produce positive/negative examples for fine-tuning the embedding model, and we use GPT-J (Wang and Komatsuzaki, 2021), a 6B-parameter LM as the scoring LM[7].

CEIL (Ye et al., 2023) proposes a compositional selection method for tn-context learning. It models the compositional interaction between the given input and in-context examples, and fine-tunes the selection model through a carefully-designed contrastive learning objective.

TST (Poesia et al., 2022) retrieves few-shot examples from a training bank using target similarity tuning. It learns to recognize utterances that describe similar target programs despite differences in surface natural language features.

In addition, we also compare with two oracle methods, in which ground truth output sequences are allowed to be leveraged for few-shot selection.

Target-KNN (oracle). We select examples with similar output embeddings. We use OpenAI Babbage and OpenAI Ada to encode the ground truth, and take the best result of two models for each task.

Target Sketch Matching (oracle). We select in-context examples with similar sketches of ground truth. For text-to-SQL tasks, we calculate the overlap of SQL key words (detailed in Appendix D.6). For COGS, we follow the target-side structural similarity setting in An et al. (2023).

4.3 Backbones and Hyper-parameters

We conduct experiments with six OpenAI language models as the backbones[8]: text-chat-davinci-002, code-davinci-002, text-davinci-003, code-cushman-002, gpt-35-turbo, and gpt-4. For generating skill-based descriptions, we always use gpt-3.5-turbo, as it is cheap and fast. We select $k = 4$ in-context examples in all experiments. We use execution-with-values accuracy[9] as the evaluation metric for text-to-SQL tasks and exact-match accuracy for COGS.

5 Main Results

Table 2, Table 3, and Table 4 report the main experimental results. We also count the number of wins, i.e., how many tasks (and sub-tasks) the method performs best on.

SKILL-KNN performs better than raw-input-based selection methods. Across all backbone models and tasks, our skill-based selections achieve the best performance among non-oracle methods. Especially, SKILL-KNN with SBERT can even outperform KNN with OpenAI embedding models. These results clearly demonstrates the necessity and effectiveness of our prompting-based rewriting. Appendix A.2 contains more experimental comparisons with existing selection methods.

SKILL-KNN performs comparable/better than fine-tuning-based method. Results in Table 4 show that SKILL-KNN can perform comparable or even better than fine-tuning-based methods. It demonstrates that optimizing the input to the embedding model can also effectively help downstream tasks without any fine-tuning.

Variants with consistency and distinctiveness can outperform the base version of SKILL-KNN. As shown in the *#Wins* column in Table 2, two variants of SKILL-KNN outperform the base version in most situations. It demonstrates that injecting consistency and distinctiveness can effectively alleviate the order sensitivity. Overall, we recommend choosing the distinctiveness variant as it wins

[7]OpenAI language models cannot be used as the scoring function since OpenAI API does not provide this functionality.

[8]In this work, the "backbone" refers to the frozen large language model for prompting.

[9]We use the official evaluation scripts for Spider in https://github.com/taoyds/test-suite-sql-eval.

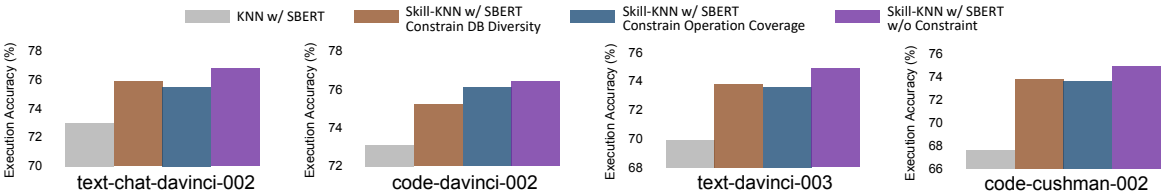


Figure 5: Performance of SKILL-KNN (base version) with constraints on selecting examples for annotating.

more times than the consistency variant (19 vs 15). When looking into detailed results, different models prefer different variants of SKILL-KNN: text-chat-davinci-002 and code-davinci-002 prefer to the distinctiveness variant, while text-davinci-003 and code-cushman-002 prefer to the consistency variant. We identify the potential factor that could lead to different preferences in Appendix B.5.

SKILL-KNN is more robust to perturbations than raw-input-based selections. Results on Dr. Spider reflect the robustness towards perturbations in data. For instance, with text-chat-davinci-002, KNN with SBERT performs lower than the random baseline on two out of three types of perturbations, while all three versions of SKILL-KNN outperform the random baseline on all three perturbations. It indicates that SKILL-KNN leads to more robust in-context learning than raw-input-based methods.

SKILL-KNN can be effective in the math reasoning task. Our study primarily evaluated the effectiveness of SKILL-KNN in the semantic parsing/code generation field. To further examine the generalizability of SKILL-KNN beyond semantic parsing, we have applied it to a challenging math reasoning task, GSM8K (Cobbe et al., 2021). Results in Table 5 evidence that SKILL-KNN can also be effective in tasks beyond semantic parsing.

6 Analysis

The most important mechanism in SKILL-KNN is the prompting-based rewriting which requires a few manually annotated demonstrations. Here we investigate how would these demonstrations affect the performance of SKILL-KNN. We experimentally analyze two factors: the number of annotated demonstrations and the selection of examples for annotation. The following analysis takes the base version of SKILL-KNN with SBERT as the embedding model and the Spider dataset for evaluation.

More annotated demonstrations bring marginal improvements. We decrease the default number of annotated demonstrations from 16 to 4/8/12. The results depicted in Figure 4 illustrate that a gradual increase in the number of annotated demonstrations can yield marginal improvements. Note that SKILL-KNN maintains the advantage compared with the raw-input-based selection even with only four annotated demonstrations. This indicates that the LLM can effectively learn how to rewrite from a limited number of examples and generalize to a wider range of unseen skills. This generalizability is also supported by our case study in Appendix C.

SKILL-KNN retains its superiority when the selection of annotation examples is constrained. We constrain the selection of annotation examples from two perspectives (detailed in Appendix D.1): first, we limit the SQL operation coverage in annotation examples; second, we restrict the annotation examples to a few databases. Figure 5 shows that applying these constraints to the selection of annotation examples leads to only a minor decline in performance, while still maintaining a substantial advantage over raw-input-based selection.

7 Related Work

In-context learning has recently become a standard paradigm for effectively leveraging large language models (Brown et al., 2020; Hendrycks et al., 2020; Patel and Pavlick, 2021; Rae et al., 2021; Zhang et al., 2022a; Hoffmann et al., 2022; Srivastava et al., 2022; Chowdhery et al., 2022; Smith et al., 2022; Wei et al., 2022a). Such a convenient paradigm has been widely applied in various scenarios such as code generation (Chen et al., 2021; Bareiß et al., 2022; Li et al., 2022b; Chen et al., 2023a; Li et al., 2023b), arithmetic reasoning (Wei et al., 2022b; Wang et al., 2022; Li et al., 2022a; Shi et al., 2023; Qin et al., 2023), and semantic parsing. From the view of leveraging skills for in-context learning, most existing work considered explicitly injecting symbolic systems into the response of the model (Cheng et al., 2023; Creswell et al., 2023; Schick et al., 2023; Shen et al., 2023;

Lu et al., 2023). This work aims to uncover the intrinsic skills from the raw inputs of examples.

Semantic parsing with deep learning methods has been explored in much existing work (Dong and Lapata, 2016; Yu et al., 2018a; Xu et al., 2017; Guo et al., 2019; Zhong et al., 2020; Wang et al., 2020; Lin et al., 2020; Scholak et al., 2021; Qi et al., 2022; Li et al., 2023a). Under the recent in-context learning paradigm, there have been some preliminary observations: Shin et al. (2021) showed that GPT-3 is better at generating English-like descriptions rather than the raw logical forms; Rajkumar et al. (2022) revealed that prompt design is essential for semantic parsing with Codex; Liu et al. (2023) showed that ChatGPT has a surprising zero-shot performance on Spider and its variants; Pourreza and Rafiei (2023) demonstrated that explicitly taking multiple stages for generating SQL leads to better in-context learning performance. These observations indicate that in-context learning has a great potential on solving semantic parsing tasks, and this work aims to further activate this potential from the view of improving few-shot selection.

Few-shot selection is one essential part for in-context learning. The standard approach is to use an off-the-shelf embedding model to encode raw inputs and select top-k similar examples (Gao et al., 2021; Liu et al., 2022; Hu et al., 2022). To improve in semantic parsing tasks, much prior work tried fine-tuning-based methods: Rubin et al. (2022) fine-tuned the embedding model based on the conditional probability under the language model; Poesia et al. (2022) trained the model to fit the target-side similarity; Hu et al. (2022) fine-tuned the model based on the similarity of between state changes; Ye et al. (2023) considered the interaction among in-context examples during training the selector. Our work points in a new direction that does not require further fine-tuning: leveraging task-specific skills by prompting large language models to rewrite input queries. Beyond semantic parsing, some more recent work tried to explore training-free selection methods for in-context learning from different perspectives: Nguyen and Wong (2023) and Li and Qiu (2023) tried to distill the whole example bank into a small set of exemplars and focused on classification tasks; Wu et al. (2023) improved classification tasks through information compression; Ye et al. (2022) and Ye and Durrett (2023) mainly focused explanation-based tasks. This work proposes prompting extremely large models to facilitate few-shot selection, which is a novel perspective to harness the power of large language models.

8 Conclusion

This work proposes SKILL-KNN to facilitate in-context learning on semantic parsing tasks. By generating skill-based descriptions without any fine-tuning, SKILL-KNN and its two variant outperform raw-input-based selections in various tasks.

Limitations

GPU resources. Our experiments have a high cost on GPU resources, since in-context learning requires extremely large language models. Specifically, all experiments are conducted on the 8 x NVIDIA A100 GPU station. During inference time, it takes about 2 x 8 GPU hours to generate for each 10,000 examples. Thus, it totally takes $400 \sim 500$ x 8 GPU hours to reproduce our Table 2.

Task type. We mainly evaluate SKILL-KNN on cross-domain semantic parsing tasks, and we believe it can also help other challenging tasks where some intrinsic task-specific skills are needed. However, for tasks that require only surface feature similarity of in-context examples, we suppose the advantage of SKILL-KNN could be diminished.

Individual variants. We design two variants of SKILL-KNN based on consistency and distinctiveness, respectively. An ideal variant should take into account both these two aspects. We take this as a future direction for our work.

Ethics Statement

Due to the use of pre-trained language models, this work can be exposed to potential ethical risks associated with general deep learning models, such as social bias and privacy breaches. We suppose this work would be helpful to alleviate potential ethical issues for in-context learning as it can better overcome the surface-form biases in example bank.

Acknowledgments

We thank all the anonymous reviewers for their valuable comments. Shengnan An and Nanning Zheng were supported in part by NSFC under grant No. 62088102.

References

Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. 2023. How do in-context examples affect compositional generalization?

Patrick Bareiß, Beatriz Souza, Marcelo d'Amorim, and Michael Pradel. 2022. Code generation tools (almost) for free? a study of few-shot, pre-trained language models on code. *arXiv preprint arXiv:2206.01335*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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 Eleventh International Conference on Learning Representations*.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2023a. Codet: Code generation with generated tests. In *The Eleventh 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, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. 2023b. On the relation between sensitivity and accuracy in in-context learning.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

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. *arXiv preprint arXiv:2204.02311*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*.

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 (Volume 1: Long Papers)*, pages 33–43.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. *arXiv preprint arXiv:2203.08568*.

Rohit J Kate, Yuk Wah Wong, Raymond J Mooney, et al. 2005. Learning to transform natural to formal languages. In *AAAI*, volume 5, pages 1062–1068.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105.

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.

Itay Levy, Ben Bogin, and Jonathan Berant. 2022. Diverse demonstrations improve in-context compositional generalization. *arXiv preprint arXiv:2212.06800*.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql.

Jia Li, Yunfei Zhao, Yongmin Li, Ge Li, and Zhi Jin. 2023b. Towards enhancing in-context learning for code generation. *arXiv preprint arXiv:2303.17780*.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023c. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Xiaonan Li and Xipeng Qiu. 2023. Finding supporting examples for in-context learning. *arXiv preprint arXiv:2302.13539*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022a. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022b. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888.

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

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training.

Tai Nguyen and Eric Wong. 2023. In-context example selection with influences. *arXiv preprint arXiv:2302.11042*.

Roma Patel and Ellie Pavlick. 2021. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *arXiv preprint arXiv:2205.06983*.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

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.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen Jr, 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*, pages 7699–7715.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. *arXiv preprint arXiv:2302.05698*.

Xi Ye and Greg Durrett. 2023. Explanation selection using unlabeled data for in-context learning. *arXiv preprint arXiv:2302.04813*.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Ves Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2022. Complementary explanations for effective in-context learning. *arXiv preprint arXiv:2211.13892*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018b. 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. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022a. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022b. Active example selection for in-context learning. *arXiv preprint arXiv:2211.04486*.

Zihao 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*, pages 12697–12706. PMLR.

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.

This is the Appendix of the paper: *Skill-Based Few-Shot Selection for In-Context Learning*.

A More Experimental Results

We report more experimental results with SKILL-KNN.

A.1 Recall@N Performance of SKILL-KNN

Table 6: Recall@N performance of SKILL-KNN with distinctiveness on Spider dev set. The backbone model is text-chat-davinci-002 and the sampling temperature is 0.7.

Top-N	1	2	3	5	7	10
Recall@N (%)	80.3	85.0	87.4	89.2	89.9	90.8

Besides the performance of greedy-decoding, we also evaluate the top-k recall performance of SKILL-KNN. Since we cannot take beam search with OpenAI interfaces, we implement the top-N selection with sampling and re-ranking, following the self-consistency setting (Wang et al., 2022). Specifically, we first sample 100 sequences and then select the k most frequently occurring sequences and evaluate their execution accuracy.

We take the text-chat-davinci-002 as the backbone model and set the sampling temperature as 0.7. We evaluate SKILL-KNN with distinctiveness on Spider dev set. Table 6 shows that the recall rate gradually goes up with increasing N and can even achieve higher than 90% when we set $N = 10$. Moreover, the top-1 in Table 6 is 80.3% which is higher than the greedy-search performance (78.3% in Table 2). It means that our SKILL-KNN can obtain further gains from ensemble methods such as self-consistency.

A.2 Comparison with More Baseline Methods

Table 7: Comparison with more baselines on Spider dev set.

Backbone	Method	Exec. Acc.
text-chat-davinci-002	Random	72.9
	Best-of-5 (Nakano et al., 2022)	73.7
	K-Center (Sener and Savarese, 2018)	73.4
	Influence (Nguyen and Wong, 2023)	73.4
	DDP (Ye et al., 2023)	74.6
	SKILL-KNN w/ SBERT (base)	76.8

Despite the baselines mentioned in Section 4.2, here we reproduce more selection methods. Results in Table 7 further demonstrate the advantage of SKILL-KNN.

B More Analysis

B.1 Motivation Behind Two Variants of SKILL-KNN

The motivation behind the two variants stems from our consideration of disturbances from the prompt-order sensitivity as additive noises to the ground-truth skills during the rewriting process. The two variants are designed to address two types of noise:

Zero-mean white noise which frequently occurs in results and originates from a zero-mean distribution (e.g., zero-mean Gaussian distribution). We assume its magnitude is relatively small compared to the ground truth. Zero-mean white noise can cause the loss or redundancy of partial information in the ground truth.

Spike noise which occasionally occurs in results and has a much larger magnitude than the ground truth. It strongly influences the information in the ground truth and causes outliers.

The consistency-based variant is more effective at addressing zero-mean white noise, as the averaging operation reduces the variance of zero-mean noise. **The distinctiveness-based variant is better suited for handling spike noise**, as it mitigates the influence of outliers. The final results in our Table 2 indicate that both types of noise occur in our LLM-based rewriting, as evidenced by the close win times of the two variants (19 vs 15).

To further support that two variants are better at tackling two different types of noise, we conduct additional analysis from two perspectives.

Table 8: The selection accuracy of different variants under different noise patterns.

	Zero-Mean White Noise	Spike Noise
Consistency	95.4%	81.9%
Distinctiveness	90.4%	87.9%

Evaluation of selection performance. We examined the performance of each variant in selecting better examples from the example bank under the two noise patterns. We construct some synthetic data for automated evaluation: we take 1,000 unique embeddings of skill-based descriptions as ground truth, then we add noise on each ground-truth embedding to construct the sample set, and

finally we assess whether each sample set could correctly select the original ground-truth embedding. The accuracies are shown in Table 8.

Human evaluation of rewriting. In Spider dev set, we first identified examples where one variant can always succeed with different LLMs while the other variant always failed. Then, we check the frequency of spike noise occurrences in these examples through human evaluation. We checked 23 examples and found that: for examples where the consistency-based variant wins, the spike noise occurs in 0.4 sample/set; for examples that distinctiveness-based variant wins, the spike noise occurs in 1.6 sample/set.

The results of the above analysis further evidence that the consistency-based variant performs better than the distinctiveness-based variant under the zero-mean white noise but performs worse under the spike noise.

B.2 The Choice of Hyper-Parameter m

Table 9: Performance with $m = 3/5/7/9$ (Dataset: Spider, LLM: code-cushman-002, variant: consistency).

m	3	5	7	9
Exec. Acc. (%)	72.8	74.7	75.1	75.0

The two variants of SKILL-KNN require to generate m candidate skill-based descriptions. In our experiments, we set $m = 5$ as a trade-off between achieving optimal performance and minimizing computational costs. During our initial exploration, we had experimented with $m = 3/5/7/9$. As shown in Table 9, the performance improves marginally when $m > 5$. Therefore, we decided to set $m = 5$.

B.3 Measuring Diversity of SKILL-KNN and Oracle methods

Table 10: Number of different databases among selected examples. This can reflect the diversity of in-context examples selected by different methods.

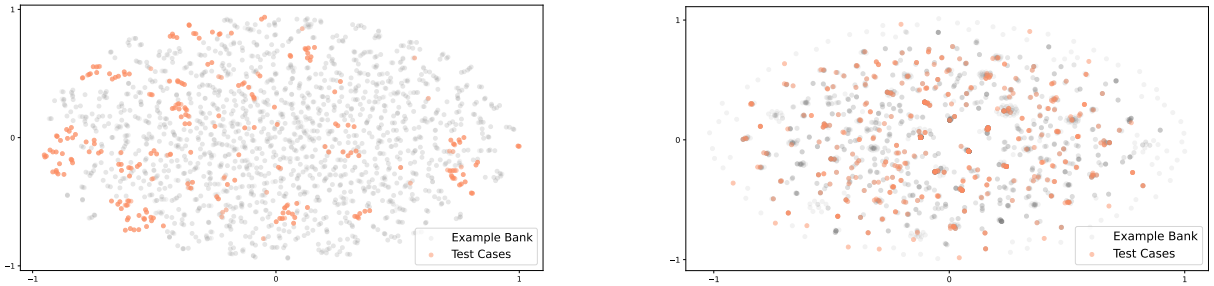
Number of Different Databases	Spider	Dr. Spider	KDBQA
SKILL-KNN (distinctiveness)	2.83	2.84	2.88
Target-KNN (oracle)	2.18	2.16	2.21

A surprising observation in Table 2 is that with the same backbone model for in-context learning, our SKILL-KNN could sometimes outperforms oracle methods. Specifically, SKILL-KNN consistently outperforms at least one oracle method on the DB sub-task in Dr. Spider and two sub-tasks in COGS (marked with underlines). Such an observation could be caused by the different diversity in selected examples. As indicated in Levy et al. (2022) and An et al. (2023), beyond the similarity to the test case, a higher diversity among in-context examples could also help better perform cross-domain generalization under in-context learning. Oracle methods directly seek higher similarity, thus the selected examples may be less diverse, which could slightly hamper the generalization performance. To reflect the diversity of in-context examples, we count the number of different databases among selected examples. Statistics in Table 10 shows that SKILL-KNN can lead to a higher diversity than oracle method, which is in line with our hypothesis.

B.4 Why do skill-based descriptions perform better?

Since both SKILL-KNN and raw-input-based methods use the embedding similarity for selection, we suppose that the higher performance of SKILL-KNN can be contributed by some desired properties in the embedding space of skill-based descriptions. Based on this inspiration, we visualize the embedding space of both raw input queries and skill-based descriptions with t-SNE (Van der Maaten and Hinton, 2008). More details are contained in Appendix D.7.

Under the embedding space of skill-based descriptions, the distribution of test cases is closer to that of the example bank, thus benefiting cross-domain generalization. For the raw-input-based embeddings of test cases, Figure 6a shows that these embeddings are mainly centralized in some local parts. On the one hand, the example bank can not be fully utilized under this embedding space, since the top-k similar examples must be around the local parts of test cases. On the other hand, the different distributions represent that this space does not reveal the inner similarity between test cases and example bank, thus is helpless to facilitate cross-domain generalization. Under the skill-based embedding space (shown in Figure 6b), the distributions of test cases and the example bank are better matched. Therefore, the cross-domain generalization gap can be better bridged with skill-based descriptions.



(a) Raw-Input-Based Embedding Space

(b) Skill-Based Embedding Space

Figure 6: T-SNE visualization for the embedding space of (a) raw input queries and (b) skill-based descriptions. The orange points are from the dev set in Spider while the gray points are from example bank.

The skill-based embedding space is more sparse, thus the boundary of similar examples is more clear and can be more robust to perturbations. As shown in Figure 6a, the raw-input-based embedding space is almost evenly distributed. It means that for one embedding in this space, the boundary to determine "which examples are similar" is not clear enough. Without a clear boundary for selecting similar examples, the performance could be non-robust to perturbations in data. Compared with the embeddings of raw inputs, the skill-based embeddings shown in Figure 6b are more clustered, thus the KNN-based selection can be more robust to data perturbations.

B.5 What factors cause the different performances of two variants?

As mentioned in Section 5, different backbone models prefer different variants of SKILL-KNN. As both two methods aim to improve skill-based similarity, such performance differences indicate that beyond similarity, some other factors also influence in-context learning. An et al. (2023) and Rajkumar et al. (2022) indicated that the complexity of selected examples could be one potential factor. Here, we explore whether the two variants differ in the complexity of the selected examples.

Consistency-based variant always leads to more complex in-context examples than distinctiveness-based variant. Here, we mainly check the complexity of in-context examples for text-to-SQL tasks from two perspectives: (1) the average number of tables, which reflects the complexity of database, and (2) the average length of SQL queries, which reflects the hardness of searching. As shown in Figure 7, consistency-based variant leads to higher complexity than distinctiveness-based variant in all tasks and under both perspectives. It can help to explain why different models prefer different variants: text-chat-davinci-002 and code-davinci-002 prefer simpler in-context examples while text-davinci-003 and code-cushman-002 are more robust to different complexity.

C Case Study

Considering that the quality of generated skill-based description can be one key factor that influences the effectiveness of SKILL-KNN, we manually check the generated skills for 100 examples. We find that 86/100 generated skills are exactly correct; 12/100 are almost correct but need some partial modifications (e.g., the number of joined tables); and only 2/100 generated skills are totally wrong.

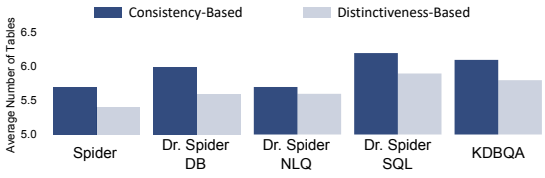
Moreover, during manually checking the quality of generated skills, we surprisingly find that there are some novel descriptions about skills that are not presented in our annotated examples. Table 11 shows some examples. It indicates that the prompting-based rewriting can provide a degree of generalization of unannotated skills.

D Detailed Settings of Experiments

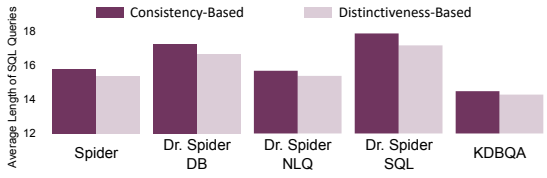
In this section, we provide more details about our experimental settings.

D.1 Select Examples for Annotation

In our default setting, we consider two principles to select examples for annotating the required skills: 1) ensuring coverage of all logical operations found in the example bank and 2) selecting examples from diverse databases. Specifically, we first find all used logical operations in the example bank and



(a) Number of Tables



(b) Length of SQL Queries

Figure 7: Two perspectives that reflect the complexity of selected in-context examples from two variants of SKILL-KNN. (a) The average number of tables contained in the database of each in-context example. (b) The average length of SQL queries (split by space).

Table 11: Case study for generated skill-based descriptions in Spider. Texts in purple are novel descriptions that are not presented in our annotations.

Input Query	Generated Skill	SQL
display the emails of the employees who have no commission percentage and salary within the range 7000 to 12000 and works in that department which number is 50.	To solve this task in the database, we need to select one column, apply an interval constraint, apply a null constraint, and apply an equality constraint.	SELECT email FROM employees WHERE commission_pct = "null" AND salary BETWEEN 7000 AND 12000 AND department_id = 50
What are the salaries and manager ids for employees who have managers?	To solve this task in the database, we need to select two columns and apply a non-null constraint on the values in another column.	SELECT salary , manager_id FROM employees WHERE manager_id != "null"
What is all the information about the Marketing department?	To solve this task in the database, we need to select all columns and apply a constraint on the values in one column.	SELECT * FROM departments WHERE department_name = 'Marketing'

greedily cover these operations in a few examples. Then, we randomly select more examples from various databases until there are 16 examples.

In our ablation study shown in Figure 5, to constrain the operation coverage, we just remove our first selection step; to constrain the database diversity, we just select examples from two databases.

D.2 Inference Hyper-Parameters

During inference, we set the max decoding length to 200, and the sampling temperature to 0.

D.3 Input-Output Formats

Figure 8 shows some input-output examples to illustrate the data formats in our experiments.

Note that the output format of COGS follows the transformation in An et al. (2023) which converts the original long-chain format into a more compact function-calling format. Such a transformation is similar to the conversion from Lambda calculus to FunQL in Geo domain(Zelle and Mooney, 1996; Kate et al., 2005; Zettlemoyer and Collins, 2012). It improves the human readability by omitting two types of details in original format: the special marker for definite descriptions and the Skolem constants. Apart from the omitted details, this transformation keeps the main semantics in the domain of COGS, such as semantic roles, modifications, and orders among clauses and modifications.

D.4 Evaluation on BIRD

Different from other text-to-SQL tasks, BIRD additionally provides "evidence" for each natural language question. Therefore, we add the evidence as part of the context for in-context learning. For evaluating raw-input-based methods on BIRD, we concatenate the natural language question and the additional evidence to compute the embedding. For our SKILL-KNN, we also provide the evidence for rewriting, and we use 12 annotated demonstrations with evidence (shown in Appendix E).

Since the database schema in BIRD is too large to be fully contained in the context for LLM, we reduce the size of schema through grounding in preprocessing. Specifically, we calculate the embedding similarity between the input question (along with the evidence) and each table name and column name. Based on this similarity, we preserve 8 tables each with 16 columns for each schema-question pair.

D.5 Evaluation on COGS

COGS totally contains 24,155 examples in train set and 21,000 examples in gen set. To reduce the high computational cost, we sample 2,000 examples from the train set as the example bank for in-context learning, and sample 1,000 examples from two sub-tasks primitive substitution (P.S.) and primitive structural alternation (P.A.) which are defined in An et al. (2023).

D.6 Target Sketch Matching for SQL

As mentioned in Section 4.2, to select in-context examples with target sketch matching (oracle) in text-to-SQL tasks, we calculate the overlap of SQL key words between the example from example

bank and the labeled SQL query of the test input query. We mainly consider the following SQL key words along with several operations for calculation: `SELECT`, `WHERE`, `GROUP`, `HAVING`, `ORDER`, `DESC`, `ASC`, `LIMIT`, `JOIN`, `INTERSECT`, `EXCEPT`, `UNION`, `NOT`, `IN`, `OR`, `AND`, `BETWEEN`, `EXISTS`, `LIKE`, `DISTINCT`, `COUNT`, `AVG`, `MIN`, `MAX`, `SUM`, `CAST`, `CASE`, `WHEN`, `THEN`, `ELSE`, `END`, `IIF`, `REAL`, `FLOAT`, `NULL`, `STRFTIME`, `*`, `/`, `=`, `>`, `,<`, `!`, `+`, `-`, `%`. Based on these key words, the target sketch similarity between two SQL queries y_t and y_i is calculated as follows,

$$\mathrm{sim}_k(y_t, y_i) = |\mathrm{KW}(y_t) \cap \mathrm{KW}(y_i)|, \quad (7)$$

in which $\mathrm{KW}(\cdot)$ returns a set of contained key words.

D.7 T-SNE Visualization

For the visualized embedding space in Figure 6, we use Sentence-BERT as the embedding model and take examples from both example bank and dev set in Spider. For SKILL-KNN, we take its consistency-based variant. To accelerate the visualization process, we just take examples with medium hardness (defined in Yu et al. (2018b)). We use the implementation of t-SNE from the sklearn library[10]. We set the learning rate of t-SNE as "auto", init method as "random", and perplexity as 3.

E Annotated Demonstrations

Table 12 lists 16 annotated demonstrations for text-to-SQL tasks and Table 13 lists another 12 annotated demonstrations with evidence (which is required in BIRD). Table 14 lists 16 annotated demonstrations for COGS. Appendix D.1 introduces how we select these examples.

[10] https://scikit-learn.org/stable/.

Table 12: Annotated demonstrations for text-to-SQL tasks. All these examples are from the example bank of Spider.

DB ID	Input Query	Annotated Skill-Based Descriptions
allergy_1	Show all majors.	To solve this task in the database, we need to select distinct values in the column.
match_season	Count the number of different colleges that players who play for Columbus Crew are from.	To solve this task in the database, we need to join two tables and count the number of distinct values in the column.
gymnast	What are the hometowns of gymnasts and the corresponding number of gymnasts?	To solve this task in the database, we need to join two tables, select one column, group these selections and count the number of selections in each group.
gas_company	Show minimum, maximum, and average market value for all companies.	To solve this task in the database, we need to return the minimum value, the maximum value, and the average of values in the column.
culture_company	Show the years, book titles, and publishers for all books, in descending order by year.	To solve this task in the database, we need to select three columns and sort them in descending order according to the values in one column.
product_catalog	Which catalog contents have a product stock number that starts from "2"? Show the catalog entry names.	To solve this task in the database, we need to select one column and apply a constraint on the format of values in this column.
bike_1	What are the dates in which the mean sea level pressure was between 30.3 and 31?	To solve this task in the database, we need to select one column and apply a constraint that the values in another column should in a certain range.
flight_1	What is the salary and name of the employee who has the most number of certificates on aircrafts with distance more than 5000?	To solve this task on the database, we need to join three tables, apply a greater-than constraint, group the selections and calculate the number of each group, sort the selections in descending order, and select the top result.
bike_1	What is the average longitude of stations that never had bike availability more than 10?	To solve this task in the database, we need to calculate the average value in one colum, apply an non-inclusion constraint with another set of selections, which need to group the selctions and find which groups have a maximum value greater than the threshold.
hr_1	display all the information of employees whose salary is in the range of 8000 and 12000 and commission is not null or department number does not equal to 40.	To solve this task in the database, we need to give full information about selections, apply an interval constraint, and apply an optional constraint that two unequal judgments should be satisfied at least one.
bike_1	What are the ids of stations that have latitude above 37.4 and never had bike availability below 7?	To solve this task in the database, we need to exclude the selections in the second set from the first set: the first set of selections need to apply a greater-than constraint, and the second set of selctions need to group the selctions and find which groups have a minimum value lower than the threshold.
bike_1	What are the names and ids of stations that had more than 14 bikes available on average or were installed in December?	To solve this task in the database, we need to return the union of two set of selections: the first set of selections need to join two tables, group the selections and find which groups have an average value greater than the threshold, and the second set of selections need to apply a constaint on the format of values.
storm_record	Show storm name with at least two regions and 10 cities affected.	To solve this task in the database, we need to return the intersection of two set of selections: the first set of selections need to join two tables, group the selections and find which groups have a number of selections greater than or equal to the threshold, and the second set of selections need to join two tables, group the selections and find which groups have a sum of values larger than or equal to the threshold.
formula_1	List the forenames of all distinct drivers in alphabetical order?	To solve this task in the database, we need to select distinct values in one column and sort these selections in ascending order according to the selected values.
hr_1	display job ID for those jobs that were done by two or more for more than 300 days.	To solve this task in the database, we need to apply a greater-than constraint on the difference between two values, group the selections and find which groups have a number of selections greater than or equal to the threshold.
small_bank_1	Find the names and total checking and savings balances of accounts whose savings balance is higher than the average savings balance.	To solve this task in the database, we need to select one column and add the values in another two columns, join three tables, and apply a greater-than constraint where the threshold is the average of another set of selected values.

For Rewriting:

Context:
Generate the needed skills to solve the task on the database schema.

Database schema:
allergy_type (allergy, allergytype)
has_allergy (stuid, allergy)
student (stuid, lname, fname, age, sex, major, advisor, city_code)
Task: Show all majors.
(Optional) ### Evidence: xxx
Skills: To solve this task in the database, we need to select distinct values in the column.

Database schema
…

Database schema:
department (department_id, name, creation, ranking, budget_in_billions, num_employees)
head (head_id, name, born_state, age)
management (department_id, head_id, temporary_acting)
Task: What are the maximum and minimum budget of the departments?
(Optional) ### Evidence: xxx
Skills: To solve this task in the database, we need to

Completion:
return the minimum value and the maximum value in one column.

For Semantic Parsing:

Context:
Generate a SQL query for the given natural language task and database schema.

Database schema:
department (Department_ID, Name, Creation, Ranking, Budget_in_Billions, Num_Employees)
head (head_ID, name, born_state, age)
management (department_ID, head_ID, temporary_acting)
Task: How many acting statuses are there?
(Optional) ### Evidence: xxx
SQL query: SELECT count(DISTINCT temporary_acting) FROM management

Database schema:
…

Database schema:
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)
Task: How many singers do we have?
(Optional) ### Evidence: xxx
SQL query:

Completion:
SELECT count(*) FROM singer

(a) Formats of Text-to-SQL Tasks (with Optional Evidence for BIRD)

For Rewriting:

Context:
Generate the required skills to parse the following sentences.

Sentence: Isabella liked that Elizabeth saw .
Skill: This sentence contains a clause in which the verb 'saw' has no object.

Sentence: A sandwich was fed to a giraffe .
Skill: This sentence is in the passive voice and has a prepositional phrase (i.e., 'to noun phrase') which describes the recipient of the verb.

…

Sentence: A rose was helped by a dog .
Skill:

Completion:
This sentence is in the passive voice and has a prepositional phrase (i.e., 'by noun phrase') which describes the agent of the verb.

For Semantic Parsing:

Context:
Please generate the semantic representations for the following sentences.

Sentence: Emma proved that a citizen offered a frog a cake on the stage .
Semantics: PROVE (EMMA , NONE , NONE) CCOMP OFFER (CITIZEN , ON (CAKE , STAGE) , FROG)

Sentence: A child attempted to smirk .
Semantics: ATTEMPT (CHILD , NONE , NONE) XCOMP SMIRK (CHILD , NONE , NONE)

…

Sentence: The cookie beside a table was shortened .
Semantics:

Completion:
SHORTEN (NONE , BESIDE (COOKIE , TABLE) , NONE)

(b) Formats of COGS

Figure 8: Input-output formats used in our experiments.

Table 13: Annotated demonstrations for text-to-SQL task with evidence. All these examples are from the example bank of BIRD.

DB ID	Input Query	Evidence	Annotated Skill-Oriented Descriptions
superstore	Please list any three orders that caused a loss to the company.	caused a loss to the company refers to Profit < 0	To solve this task in the database, we need to select one column, apply a less-than constraint, and return three results.
disney	Calculate the percentage of voice actors whose main character in the movie is in the Drama genre.	DIVIDE(COUNT(voice-actor where genre = 'Drama'), COUNT(voice-actor)) as percentage;	To solve this task in the database, we need to join three tables and calculate a percentage number. Additionally, to calculate the percentage number, we need to count the number of values with an equivalent constraint, cast the count into a real number, multiply this number by 100, and divide it by the another count.
legislator	Among the legislators who will end in 2009, how many are from the Republican party?	the legislators who will end in 2009 refers to END 2009; from the Republican party refers to party = 'Republican'	To solve this task in the database, we need to select two columns, apply an equivalent constraint on the time and an equivalent constraint on the text value.
works_cycles	Calculate the average length of employment for employee working in the Research and Development deparment.	average length of employment = AVG(subtract(2022, year(HireDate)))	To solve this task in the database, we need to get two times, calculate the differences between two times, and calculate the average of these differences. Additionally, we need to join three tables and apply an equivalent constraint.
retail_complains	Among the teenager clients who use Google account and Microsoft account, which group of client is more than the other?	teenager refers to 13 < age < = 19; Google account refers to email like '%@gmail.com'; Microsoft account refers to email like '%@outlook.com'	To solve this task in the database, we need to compare the number of values in two formats, and return the value that has a higher number. Additionally, we need to apply a between-and constraint.
university	What are the top three universities with the most international students?	most international students refers to MAX(SUM(DIVIDE(MULTIPLE(pct_international_students, num_students), 100))); name of university refers to university_name;	To solve this task in the database, we need to select distinct values from one cloumn, join two tables, group the results, order the results in descending order according to the sum of values, and return the top three results. Additionally, during getting the sum of values, we need to multiply the values in one column with percentages in another column and divide the results by 100.
airline	What is the percentage of flights which landed at Pittsburgh were faster than scheduled?	percentage = MULTIPLY(DIVIDE(SUM(ACTUAL_ELAPSED_TIME < T2.CRS_ELAPSED_TIME), COUNT(Code)), 100); landed at refers to DEST; Pittsburgh refers to Description which contains 'Pittsburgh'; faster than scheduled refers to ACTUAL_ELAPSED_TIME < CRS_ELAPSED_TIME;	To solve this task in the database, we need to compare values in two columns and convert the comparison result into a percentage. Additionally, we need to join two tables, apply a constraint on the format of values in one column, and ensure that the values in two columns are not null. Moreover, to get the percentage number, we need to cast the sum of values into a real number, multiply this number by 100, and divide it by the total count.
retail_complains	Between 1/1/2017 and 4/1/2017, what is the average server time of calls under the server DARMON?	between 1/1/2017 and 4/1/2017 refers to Date received between '2017-01-01' and '2017-04-01'; average server time refers to avg(ser_time)	To solve this task in the database, we need to calculate the average of the selected values and apply a between-and constraint. Additionally, to obtain the times from values in text format, we need to extract substrings from these texts and cast them into real numbers.
soccer_2016	When did Chennai Super Kings play its first match?	match date refers to Match_Date; Chennai Super Kings refers to Team_Name = 'Chennai Super Kings'; first match refers to min(Match_Date)	To solve this task in the database, we need to apply an either-or constraint, sort the selected results in ascending order, and return the top one result.
retails	Which ship mode has more "deliver in person" instructions, rail or mail?	ship mode refers to l_shipmode; "deliver in person" instruction refers to l_shipinstruct = 'DELIVER IN PERSON'	To solve this task in the database, we need to count the number of two values in ome column and return the value with a larger count. Additionally, we need to apply an equality constraint.
cookbook	Which ingredient appeared the most in recipes? Calculate its amount of appearance in percentage.	ingredient appeared the most in recipes refers to MAX(COUNT(ingredient_id)); calculation = MULTIPLY(DIVIDE(COUNT(MAX(ingredient_id)), COUNT(ingredient_id)), 100)	To solve this task in the database, we need to select one column and calculate one percentage number, join two tables, group the selected results, and sort the results in descending order according to the size of each group. Additionally, to calculate the percentage number, we need to cast the count of vaules into a float number, multiply this number by 100, and divide it by the another count.

Table 14: Annotated demonstrations for COGS. All these examples are from the example bank of COGS.

Input Query	Annotated Skill-Based Descriptions
Isabella liked that Elizabeth saw .	This sentence contains a clause in which the verb 'saw' has no object.
A sandwich was fed to a giraffe .	This sentence is in the passive voice and has a prepositional phrase (i.e., 'to noun phrase') which describes the recipient of the verb.
Benjamin froze .	The verb 'froze' has no object.
Sophia was given a cookie by Emma .	This sentence is in the passive voice, has an object and has a prepositional phrase (i.e., 'by noun phrase') which describes the agent of the verb.
Sophia liked a box on the cake .	This sentence has a single object with a modification phrase.
Emma sold the drink beside a road to a zebra .	This sentence has a direct object with a modification phrase and has a prepositional phrase (i.e., 'to noun phrase') which describes the recipient of th verb.
A lion ate .	The verb 'ate' has no object.
Eleanor was offered the ball .	This sentence is in the passive voice and has an object.
A box was helped .	This sentence is in the passive voice and has no object or prepositional phrase.
The fish dreamed to walk .	This sentence has an infinitive verb.
A cat lended a lawyer the cake .	This sentence has an indirect object and a direct object.
The basket was handed to a cat by Emma .	This sentence is in the passive voice and has two prepositional phrases: the first one (i.e., 'to noun phrase') describes the recipient of the verb and the second one (i.e., 'by noun phrase') describes the agent of the verb.
Sofia thought that a pancake rolled .	This sentence contains a clause in which the verb 'rolled' has no object.
Liam hoped that the boy wanted to dance .	This sentence contains a clause that has an infinitive verb.
The cat gave Ethan a rose on the table .	This sentence has an indirect object and a direct object with a modification phrase.
The duke was passed the shell on a table in the house by Emma .	This sentence is in the passive voice, has an object with a nested modification phrase, and has a prepositional phrase (i.e., 'by noun phrase') which describes the agent of the verb.