An Integrated Interactive Framework for Natural Language to SQL Translation

Yuankai Fan[1(✉)], Tonghui Ren[1], Dianjun Guo[1], Zhigang Zhao[1,2], Zhenying He[1], X. Sean Wang[1], Yu Wang[3], and Tao Sui[3]

[1] Fudan University, Shanghai, China
{fanyuankai,djguo20,zhenying,xywangcs}@fudan.edu.cn, thren22@m.fudan.edu.cn
[2] Shandong Computer Science Center (National Supercomputer Center in Jinan), Jinan, Shangdong, China
zgzhao21@m.fudan.edu.cn
[3] China UnionPay Merchant Services Co., LTD., Shanghai, China
{wangyu62,taosui}@chinaums.com

Abstract. Numerous web applications rely on databases, yet the traditional database interface often proves inconvenient for effective data utilization. It is crucial to address the significant demand from a vast number of end users who seek the ability to input their requirements and obtain query results effortlessly. Natural Language (NL) Interfaces to Databases (NLIDBs) with interactive query mechanisms make databases accessible to end users and simultaneously retain user confidence in the results. This paper proposes an approach called IKnow-SQL for building interactive NLIDBs. IKnow-SQL introduces a unified framework for translation models to improve accuracy and increase interactivity. Specifically, IKnow-SQL first employs an underlying translation model to parse the semantics of a given NL query. By evaluating the model behavior, IKnow-SQL then recognizes the parts of the model output that may require human intervention. Next, IKnow-SQL presents clarifying questions to solicit and memorize user feedback until a polished result is obtained. Extensive experiments are performed to study IKnow-SQL on the public benchmark. The results show that the translation models can be effectively improved using IKnow-SQL with less user feedback.

Keywords: Interactive Semantic Parsing · Database System · NL2SQL

1 Introduction

In the modern digital landscape, web applications have become an integral part of our daily lives. These applications provide us with a wide range of functionalities, from social networking and e-commerce to productivity tools and content management systems. However, behind the scenes, the smooth functioning of these web applications often hinges on the effective utilization of databases for storing, managing, and retrieving data [9,15].

F. Zhang et al. (Eds.): WISE 2023, LNCS 14306, pp. 643–658, 2023.
https://doi.org/10.1007/978-981-99-7254-8_50

The concept of Natural Language Interfaces to Databases (NLIDBs) has emerged as a visionary approach to enhance database accessibility for a wide range of non-technical users. The ultimate goal is to enable users to interact with databases in a manner that closely resembles human conversation. Recent advancements in natural language processing technologies, particularly neural machine translation, have rekindled research interest in developing NLIDBs [3,4,19,20,26,28,33]. The main idea is to consider the NL to SQL translation (NL2SQL) problem as a translation task and train a generalized sequence-to-sequence (Seq2seq) model in a supervised manner.

Despite the significant gains in terms of translation accuracy, the translation may still exhibit inaccuracies or errors, particularly observed in some complex queries [8]. As a solution, *interactive NLIDBs* have been further proposed as a promising paradigm [14,36], which includes human users in the loop to resolve utterance ambiguity, boost system accuracy, and improve user confidence via human-machine collaboration [5,7,21]. While many recent studies successfully demonstrated the value of interactive NLIDB in practice, the interaction is for an individual prediction, and thus the feedback users provide is not preservable. Namely, if users input the same NL query again, a repeated interaction may be triggered with no change. Consider the example in Fig. 1 that shows two-way communication between an end-user and the NLIDB system during two interactive sessions. While the system has the capability to detect the confusion span (i.e., *"under age 30"*) in the input queries that need clarification, the two sessions request repeated feedback due to the same confusion span being recognized. This example shows that an inefficient interaction may be involved in the current interactive NLIDB system.

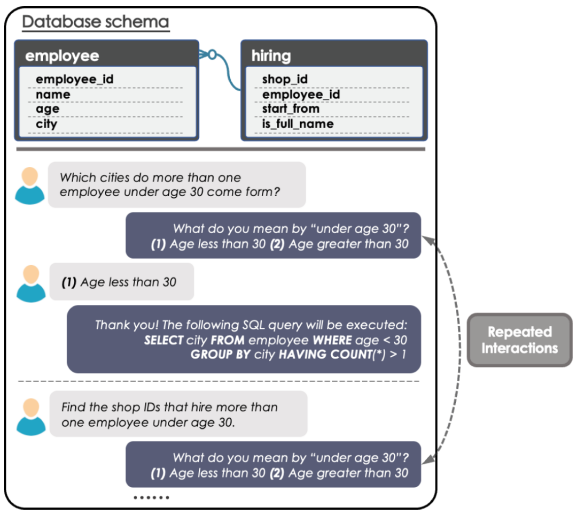

Fig. 1. An example of repeated interaction exists in the current interactive NLIDB. The database and queries are from the SPIDER benchmark [38].

In this paper, we present IKnow-SQL, an **I**nteractive **Know**ledge preservation approach for NL2**SQL** translation. IKnow-SQL is inspired by the cache mechanism used in deep learning [12,31] and aims to minimize human feedback while maximizing translation accuracy by storing and recalling knowledge gained through human interactions.

Concrete speaking, IKnow-SQL belongs to the field of *interactive semantic parsing* in natural language processing (NLP) and provides a **unified framework** for any back-end NL2SQL translation model that can interact with users. Given an NL query, IKnow-SQL performs the following steps: ① uses an underlying translation model for semantic parsing and keeps track of its internal states during decoding; ② employs a confidence estimator to identify parts of the model output that may require human intervention based on probability mass at each decoding step; Assuming human intervention is needed, ③ IKnow-SQL generates questions for user clarification using a question generator and memorizes user feedback through a knowledge collector until no low level of confidence is detected. Here, the mechanism for knowledge memorization in this context stores information from past interactions, which allows for a reduction in the number of user interactions needed over time.

To better demonstrate the advantages of IKnow-SQL, we conduct our experiments on the popular NL2SQL benchmark Spider [38], by applying to three base NL2SQL models [4,20,28]. We empirically verified that with a small amount of targeted, test time user feedback, IKnow-SQL can significantly improve the accuracy of base semantic parsers by 8.1% to 15.4% absolute.

To summarize, our contributions are three-fold:

- We propose a unified framework IKnow-SQL for building interactive NLIDBs, which can be used for any existing NL2SQL translation models to improve the performance and efficiency.
- By incorporating a knowledge caching mechanism in IKnow-SQL, a same run-time accuracy of a base semantic parser can be retained with soliciting less user feedback.
- We perform a series of experiments to evaluate IKnow-SQL on the public NLIDB benchmark with three state-of-the-art Seq2seq-based translation models. The experimental results demonstrate the effectiveness of IKnow-SQL for the NL2SQL problem.

The remainder of this paper is organized as follows. We first discuss the related works of IKnow-SQL in Sect. 2. We then give an overview of IKnow-SQL in Sect. 3. Next, we present the details of each key components used in IKnow-SQL in Sect. 4. Finally, we presents the results of our experiments on IKnow-SQL in Sect. 5 and conclude in Sect. 6.

2 Related Works

Natural Language Interface to Database. NLIDBs have been studied for several decades in database management and NLP communities. Early works [1,

2,18,25,27,29,39] are rule-based approaches, which use handcrafted grammars and rules to map NL queries to SQL queries specific to a certain database.

The recent success of deep learning, notably in the realm of neural machine translation, have catalyzed the adoption of machine learning-based approaches to build NLIDB systems. These approaches treat the NLIDB challenge as a Seq2seq translation task and employ the encoder-decoder neural architecture to do the translation [3,4,13,20,33,34,37]. In addition, with the increasing attention given to large-scale language models (LLMs) [23,24,32] recently, they have emerged as a new approach for the NL2SQL task. In contrast to conventional end-to-end SQL query generation, IKnow-SQL employs an interactive mechanism to achieve more precise NL2SQL translation, thereby enhancing the accuracy of the resulting SQL queries.

Interactive Semantic Parsing. Interactive natural language interfaces for databases are a promising solution to improve translation accuracy as well as user confidence in practical applications [6,14,16,18,30,35]. However, current solutions are limited to specific formal languages and datasets, making them somewhat ad-hoc. The design of IKnow-SQL centers around creating an interactive NLIDB framework independent of any specific model. Moreover, IKnow-SQL is dedicated to integrating external knowledge into the system, aiming to minimize the extent of user feedback required.

3 IKnow-SQL Overview

Figure 2 shows a high-level view of IKnow-SQL. The main process unfolds as:

1. When a user query is fed into the base NL2SQL model, the model parses its semantics and then generates a corresponding raw SQL query initially.
2. The raw SQL query, along with the probability mass at each SQL token, is estimated by the confidence estimator to determine if any unconfidently-predicted target tokens exist.
3. If any uncertainties detect, the question generator is used to interact with users by generating a series of clarifying questions to solicit user input.
4. The user feedback is then used to update the states of the NL2SQL model and then generate a new polished SQL query for the user.
5. Meanwhile, the knowledge collector will also be incorporated to memorize the user feedback and sync with the underlying NL2SQL model.

Among these, incorporating the knowledge collection process into the semantic parsing is unique to our setup. We describe each above step below.

Semantic Parsing. This step in Fig. 2-① is to utilize an underlying NL2SQL model to parse the NL query that is given by the user to the SQL query counterpart. Mapping NL queries to their formal semantic representations (i.e., SQL queries), the mainstream models usually formulate the NL2SQL problem as a language translation task, and employ the Seq2seq framework to build the models [3,4,20,28,33]. These models can be recurrent neural network (RNN)-based simple encoder-decoder networks or the advanced attention-based encoder-decoder RNN or state-of-the-art transformer models. Concretely speaking,

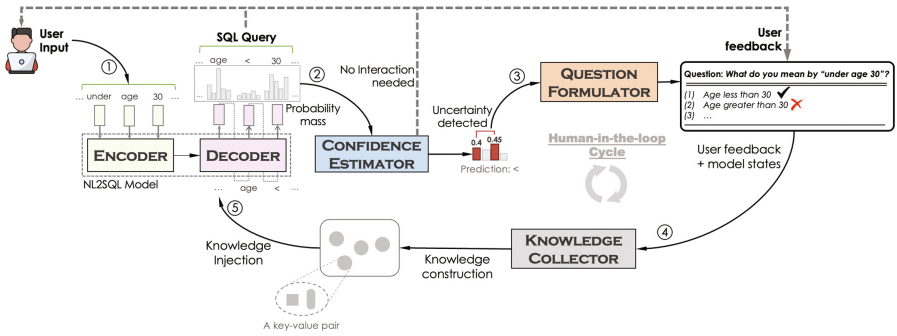

Fig. 2. A diagram illustrating the workflow of IKnow-SQL.

taking a given NL query and a corresponding database schema, the model uses an encoder to compute a contextual representation by jointly embedding the NL utterance with schema information. Afterward, an auto-regressive decoder is used to compute a distribution over the SQL programs. Depending on different model designs, the learning target of the decoder can be raw SQL tokens [20,40], intermediate representations of SQL language [13], or SQL AST trees [4,28,33]. The NL2SQL model is generally trained with maximum likelihood estimation.

For example, assume the first NL query "*Which cities do more than one employee under age 30 come from?*" in Fig. 1 is given, an NL2SQL model may interpret it to a SQL query as below,

```
SELECT city FROM employee
WHERE age > 30
GROUP BY city HAVING count(*) > 1
```

The SQL query may be incorrect and will undergo subsequent processing steps.

Prediction Inspecting. The step in Fig. 2-② is an *inspection process* where IKnow-SQL examines the state of each decoding step of the underlying NL2SQL model and determines if any error (i.e., unconfidently-prediction) exists in the SQL query output, and hence decides whether human intervention is needed. For example, after the underlying NL2SQL model outputs the above SQL query, IKnow-SQL introspects its states at each decoding step and recognizes a lack of confidence in the prediction concerning the `WHERE` clause (i.e., `WHERE age > 30`). This prompts IKnow-SQL to solicit user feedback on the unconfidently-prediction to refine the initial SQL query output.

User Interaction. If any unconfidently-prediction is detected, the step in Fig. 2-③ is triggered to generate the clarification question for the user. IKnow-SQL formulates as an NL generation task and leverages a template-based method to customize the question. For example, by clarifying the unconfidently-prediction `WHERE age > 30`, IKnow-SQL generates the following clarification question with two options accordingly,

Please select any options that the system needs to enforce as condition:
(1) Age of employee is large than 30.
(2) Age of employee is less than 30.

The type of the clarification questions in IKnow-SQL are fixed (i.e., multi-choice). However, it can be extended with more human-curated templates to accommodate more complex interactions.

Knowledge Memorization. To preserve the knowledge the interactions obtain from the users, we explore an approach at the step in Fig. 2-④ to memorize the knowledge properly. In IKnow-SQL, each knowledge concept is modeled as a *key-value pair* (k, v), where the key k is the combination of both the predicted token and the input tokens that have top-K attention probabilities, and the value v is the answer that the user provides; All the knowledge concepts are then stored and managed in the knowledge bank (e.g., MongoDB). IKnow-SQL therefore can leverage the knowledge to eliminate model uncertainties without human intervention. We posit that less human intervention can be avoided with more and more knowledge modeled and stored in the knowledge bank.

4 IKnow-SQL

This section provides a detailed explanation of IKnow-SQL design. We begin by explaining the confidence modeling technique, then describe the question generation process. Finally, we discuss the knowledge caching mechanism.

4.1 Difference-Based Confidence Modeling

The main observation of the inspection step is that a one-pass forward process while decoding may involve some uncertainties that the NL2SQL model likely makes mistakes. Therefore, an essential question for IKnow-SQL is how to detect the potential errors a generated sequence has. We follow the confidence estimation techniques that appeared in the literature [10,17,22], and explore an approach to estimate model confidence at each token position.

Intuitively if the underlying NL2SQL model gives similar output values over several top predictions at a decoding step, the NL2SQL model is likely uncertain about the prediction (i.e., the low-confidence prediction). Without losing generality, we assume that the decoder of the base NL2SQL model is implemented with a vanilla recurrent neural network (RNN), and illustrate the process below.

During inference, the decoder employs the RNN network to unroll the target information. At the j-th step, the target hidden state h_j is given by

$$\boldsymbol{h}_j = \mathbf{RNN}(\boldsymbol{e}_{y_{j-1}}, \boldsymbol{h}_{j-1}, \boldsymbol{c}) \tag{1}$$

At the j-th step, a probability distribution P_j is created for all possible target vocabulary symbols. This is done by considering the embedding of the previously predicted symbol, the representation of the source context, and the hidden state,

$$\begin{aligned} \boldsymbol{t}_j &= g(\boldsymbol{e}_{y-1}, \boldsymbol{h_j}, \boldsymbol{c}) \\ \boldsymbol{l}_j &= \mathbf{W}_o(\boldsymbol{t}_j) \\ P_j &= \text{softmax}(\boldsymbol{l}_j) \end{aligned} \tag{2}$$

where g denotes a linear transformation, W_o is used to map t_j to logits l_j representing the raw (non-normalized) predictions for each target symbol. These logits are subsequently normalized into the output probabilities using a softmax function[1]. Thus, the 1-best symbol is selected as the predicted symbol,

$$y_j = \text{argmax}(P_j) \tag{3}$$

Table 1. The templates used in IKnow-SQL

Templates
Does the system needs to return all information about table? → Keyword[∗]
Please select any options from the following list that the system needs to enforce as tables? → *table*
Please select any option that the system needs to enforce as conditions to join the two tables? → (*table.col*, Op, *table.col*)
Please select any option that the system needs to enforce as conditions? → (*table.col*, Op, *table.col*|Val)
Please select any option that the system needs to enforce as conditions? → (*table.col*, Op, Val)
Please select any option that the system needs to enforce group items with? → *table.col*
Please select any options that the system needs to enforce as a mathematical calculations when group items? → (Agg, *table.col*)
Please select any option that the system needs to enforce as conditions when group items? → (*table.col*, Op, *table.col*|Val)
Please select any option that the system needs to enforce return information about? → *table.col*
Please select any option that the system needs to enforce as order when sort the results? → *table.col*
Please select any options from the following list that the system needs to enforce as order when sort the results? → Keyword[*desc*|*asc*]
Please select any options from the following list that the system needs to enforce as number of results to return? → Val

Since the softmax probabilities are computed with the fast-growing exponential function, minor additions to the softmax inputs, i.e., the logits, can lead to substantial changes in the output distribution, the prediction probability from a softmax distribution has a poor direct correspondence to confidence [17]. Thus, to gauge confidence, we employ the raw predictions (i.e., logits) to estimate the j-th step output by using an indicator function $\mathbb{L}$,

$$\mathbb{L}_{y_j} = \begin{cases} 0 \text{ if } \mid l_j^{top_1} - l_j^{top_2} \mid < \omega \\ 1 \text{ otherwise} \end{cases} \tag{4}$$

where $l_j^{top_1}$ and $l_j^{top_2}$ are the 2-highest logits, and ω is a threshold. That is, if the difference between the 2-highest logits is less than the threshold, the prediction at the j-th step is modeled as low-confidence.

Probability-Based Confidence Modeling. In addition to the difference-based confidence measure, a more straightforward approach is to use the predicted probability to serve as the model confidence [17,36]. Namely, a prediction o_t needs user clarification if its probability is lower than a threshold p^*, i.e.,

$$p(o_t) < p^* \tag{5}$$

We also experiment IKnow-SQL with this confidence measure in Sect. 5.

[1] In accordance with [11], model predictions are the weighted sum of target embeddings over output probabilities.

4.2 Template-Assisted Question Generation

The goal of the clarification question generation is to synthesize an NL expression corresponding to the generated SQL query, particularly the low-confidence situations that the confidence estimator detects. In order for users to better understand the potential error IKnow-SQL needs to clarify, We follow the method introduced in [36] to design the question formulator in IKnow-SQL. In this section, we provide a short description of the approach in [36] (we call the method Rule-qg in the rest of the paper) and then explain the improvements needed for the purpose of IKnow-SQL.

The core idea of Rule-qg is to generate the clarification questions based on the SQL components that the unconfident-predictions belong to. Specifically, Rule-qg is a rule-based method, which pre-defines a seed lexicon and a set of templates for deriving questions. We adopt the idea and design our seed lexicon and templates, respectively:

- The seed lexicon defines NL descriptions for basic SQL elements in the form of $n \rightarrow t[p]$, where n is an NL phrase, t is a pre-defined syntactic category, and p is either an aggregator (e.g., `avg`) or an operator (e.g., >). For example, "*under* $\rightarrow$ Op[>]" defines the word "under" to describe the operator ">". Rule-qg considers four syntactic categories: Agg for aggregators, Op for operators, Col for columns and Q for generated questions.
- The templates define the production rules to derive clarification questions. Rules associated with each Q-typed item "$Q[v||\texttt{Clause}]$" constructs an NL question asking about v in `Clause`. The `Clause` represents a specific SQL component (e.g., **`WHERE`** `age > 30`), which is the necessary context to formulate meaningful questions.

IKnow-SQL follows the methods introduced in Rule-qg but made some changes: (1) We categorize templates based on their associated `Clauses`, and simplify the derivation of questions by directly instantiating the NL expressions using `Clause`. (2) Instead of constructing simple binary questions, we leverage the pre-defined seed lexicon and the semantics of v to construct multi-choice questions. Table 1 shows the templates used in IKnow-SQL.

4.3 Attention-Based Knowledge Acquisition

Once IKnow-SQL received the user feedback from the interface of the question formulator, an important question for IKnow-SQL is how to memorize the user feedback and make it preservable for the system.

Typically, storing the NL query that the user interacts with associates with the user feedback can effectively avoid repetitive interactions as shown in Fig. 1. Although intuitive, this hard one-one mapping memorization suffers from low generalizability and is extremely sensitive to language variations. Therefore, finding a more generalized way to memorize external knowledge is indispensable.

Motivated by recent works of the cache mechanism used in deep learning [12,31], we introduce an attention-based method to store the external knowledge

obtained from users. The core idea of this method is to exploit the attention distribution over input tokens at a low-confidence step to define the knowledge. Without losing generality, we assume that the interaction is triggered when predicting t-th target token, in the sense that the user feedback is solicited for clarifying t-th prediction.

Specifically, we use key-value pairs to define each knowledge concept, in which the key is constructed with the following two parts,

- **Predicted token.** Since the underlying NL2SQL model is hesitant on predicting t-th token, we directly use the predicted token as part of the key.
- **Top-K input tokens.** Different token contributions vary for a specific prediction. Hence, we rank the input tokens based on the attention distribution over t-th prediction and select the tokens with top-K probabilities as another part of the key. For instance, Fig. 3 shows an example of the attention distribution of the predicted SQL query over the given input query (i.e., *Which cities do more than one employee under age 30 come from?*). For the specific "$>$" token, the corresponding top-5 input tokens are "*30*", "*than*", "*under*", "*one*", "*more*".

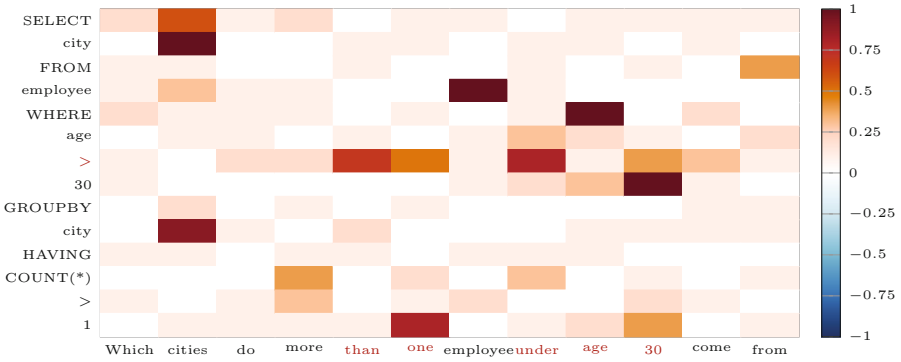

Fig. 3. An example of the attention distribution of BRIDGE model over the given NL query. The token with darker color indicates a higher probability. (Color figure online)

On the other hand, we use the answer that the user selects in the clarification question to serve as the value of the pair.

5 Experimental Evaluation

In this section, we evaluate the effectiveness of IKNOW-SQL by applying it to the state-of-the-art NL2SQL models and analyzing the experimental results.

5.1 Experimental Setup

Benchmark. We use the public SPIDER benchmark [38] to evaluate the performance of IKNOW-SQL. **Spider** is a state-of-the-art, large-scale benchmark for complex and cross-domain NL2SQL tasks. The benchmark includes 10,181 NL queries and 5,693 unique complex SQL queries on 206 databases with multiple tables covering 138 different domains. Based on the hardness levels it defined, SPIDER authors split the benchmark data into four types, namely *Easy*, *Medium*, *Hard*, and *Extra Hard*. Unlike other existing NLIDB benchmarks, SPIDER uses different databases in train and validation data sets. That is, a database schema is used exclusively for either training or validation, but not both.

Since the test set of SPIDER is hidden behind an evaluation server, the experiments we perform are on the validation set of the SPIDER benchmark.

Evaluation Metrics. We measure the query *exact-match accuracy* (Acc_{em})[2] and the *effective interaction rate* ($Rate_{ei}$) in our experiments.

Exact-Match Accuracy. If the generated SQL query exactly matches the "gold" SQL after normalization, then the translation is said to be accurate. It is a performance lower bound since a semantically correct SQL query may differ from the "gold" SQL query syntactically. This metric is the same as the *Exact Match Accuracy* metric suggested by SPIDER.

Effective Interaction Ratio. To assess the efficiency of the confidence estimation (i.e., error detection) process in IKNOW-SQL, we measure the number of effective interaction rounds. This is done by determining the ratio of the actual number of errors detected by the error detection method to the total number of detected errors. Each time a false alarm is triggered, there will be an interaction with the user, and this metric reflects the number of such interactions.

NL2SQL Models. We apply IKNOW-SQL to the following three state-of-the-art machine learning-based NL2SQL models:

- BRIDGE [20] represents the question and schema in a tagged sequence where a subset of the fields are augmented with cell values mentioned in the question.
- GAP [28] is a pre-trained model that was built upon RAT-SQL [33]. The RAT-SQL model is a schema transformer-based model that is relation-aware, and it utilizes a self-attention mechanism to enhance the encoding capacity.
- LGESQL [4] a line graph enhanced text-to-SQL model to mine the underlying relational features without constructing metapaths.

5.2 Experimental Results

We first present the results of the simulation evaluation over the public benchmark, and then we report a user study to assess the effectiveness of IKNOW-SQL.

[2] We use the terms exact-match accuracy and translation accuracy interchangeably.

Table 2. The simulation evaluation of IKnow-SQL on the validation set of Spider. "IKnow-SQL$^{\omega=X}$" denotes IKnow-SQL with difference-based confidence estimation approach (threshold at ω).

NL2SQL Model	Bridge [20]		Gap [28]		Lgesql [4]	
	Acc_{em}	$Rate_{ie}$	Acc_{em}	$Rate_{ie}$	Acc_{em}	$Rate_{ie}$
w/o interaction	70.0	–	71.8	–	75.0	–
IKnow-SQL$^{\omega=2}$	74.3	67.7	73.2	63.2	84.9	66.7
IKnow-SQL$^{\omega=3}$	76.0	**68.8**	75.8	64.3	86.4	67.2
IKnow-SQL$^{\omega=4}$	78.1	68.2	77.1	64.1	88.3	62.8
IKnow-SQL$^{\omega=6}$	**80.5**	67.2	79.9	60.8	90.4	49.2

Simulation Evaluation. In simulation evaluation, each model interacts with a simulated user, who always gives the correct answer based on the ground-truth SQL query for each clarification question. If the model fails to generate the ground truth in three consecutive interaction turns, then it is regarded as an error for that particular turn.

The results of all the models on the validation set of the Spider benchmark are presented in Table 2, both with and without the application of IKnow-SQL. These results demonstrate that IKnow-SQL can significantly enhance the overall performance of the three NL2SQL models, and the magnitude of this improvement becomes more noticeable as the threshold ω value increases. Notably, IKnow-SQL provides a significant performance boost to the Bridge model, increasing its exact match accuracy from 70% to 80.5% at threshold 6, and obtaining 68.8% effective interaction rate at threshold 3.

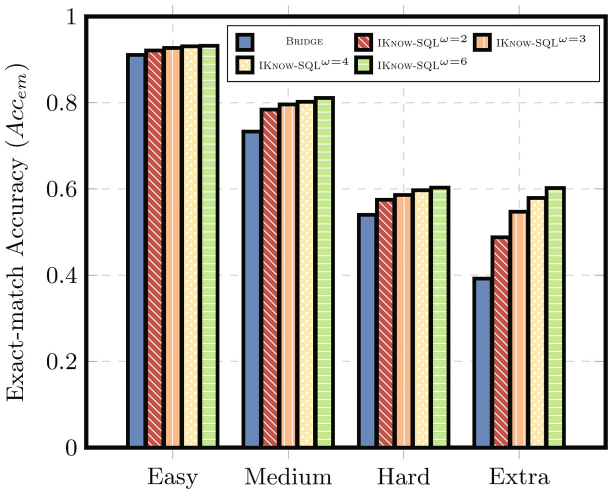

Fig. 4. Breakdown results of Bridge model on the validation set of Spider.

Next, we carry out further experiments on IKnow-SQL by applying to Bridge. Figure 4 displays the translation accuracy breakdown for different SQL difficulties on the Spider validation set. The results reveal that the performance of Bridge has improved across all four categories. In particular, when using IKnow-SQL with the "$\omega = 6$" setting, Bridge achieves an exact-match accuracy of 93.2% for the "Easy" queries (248 out of 1034), and a 60.2% accuracy for the "Extra Hard" queries (164 out of 1034 queries). We believe that since Bridge can perform well on simpler queries, IKnow-SQL can significantly enhance its performance on more complex ones. However, there is only a minimal improvement observed when IKnow-SQL is applied to Bridge. The reason behind this may be the fact that Bridge designs with a simple sequence decoder that does not inherently support nested queries. This issue is also evident in the nested queries within the "Extra Hard" category.

In order to assess the efficacy of the confidence estimator module in IKnow-SQL, we compare the performance of IKnow-SQL with Bridge with a probability-based approached, as we discussed in Sect. 4.1. The results are shown in Table 3, and they indicate that IKnow-SQL's difference-based approach outperforms the probability-based method in terms of both translation accuracy and effective interaction rate. These findings illustrate the effectiveness of the difference-based approach within the interactive paradigm.

Table 3. Evaluation on two different confidence estimation methods.

Metrics	IKnow-SQL				IKnow-SQL$^{p=X}$	
	$\omega = 2$	$\omega = 3$	$\omega = 4$	$\omega = 6$	$p = 0.95$	$p = 0.8$
Acc_{em}%	74.3	76.0	78.1	**80.5**	**77.1**	73.8
$Rate_{ei}$%	67.7	**68.8**	68.2	67.2	52.7	**68.2**

5.3 User Study

In the rest of this section, we report a human user study to assess IKnow-SQL. Our evaluation setting follows [36]: We randomly sampled 20 NL-SQL examples from the Spider validation set, and asked ten computer science students who have knowledge of SQL to task on each query with IKnow-SQL. Firstly, participants were briefed about the study and then used a toy query to demonstrate the whole process. We recorded the results of each participant for the given queries.

Figure 5 illustrates that IKnow-SQL can correctly translate most of the "Easy" queries (an average of 18 out of 20 queries) as reported by the participants. However, the participants reported difficulties with the "Hard" queries and the interactions with IKnow-SQL were not very helpful in those cases. Furthermore, we conducted brief interviews with each participant, and one participant expressed confusion about certain words returned by IKnow-SQL during the interactions, even though the natural language interface made it easier to query and interact with the underlying database.

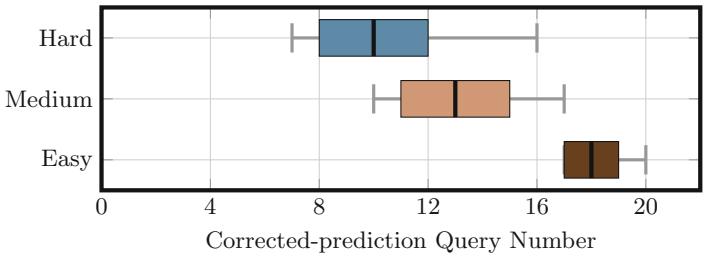

Fig. 5. Box plot of corrected-prediction count for different difficulties of queries.

The research findings indicate that an interactive natural language interface can enhance user interactions with databases. Nevertheless, additional improvements might be needed to effectively manage more intricate queries.

5.4 Error Analysis

To better understand IKnow-SQL, we examine the failed cases and analyze the potential reason. We identify the following two major causes for the failures.

- **Clause Existence Problem.** One of the major failures is caused by the limitation of the implementation of the confidence estimator module. That is, IKnow-SQL in the current setting can only detect the potential errors that occurred in a specific SQL clause, it is unable to detect if the potential error is for the existence of a certain clause. For instance, an example in Spider,

  NL Query: *Find name of the employee who got the highest one time bonus.*
  Predicted SQL of IKnow-SQL:
  ```
  SELECT employee.name
  FROM employee JOIN evaluation
  GROUP BY employee.name
  ORDER BY evaluation.bonus DESC LIMIT 1
  ```
  Gold SQL:
  ```
  SELECT employee.name
  FROM employee JOIN evaluation
  ORDER BY evaluation.bonus DESC LIMIT 1
  ```

  The `GROUP BY` clause is unnecessary for this prediction. Such failures may be avoided if finding a way to enhance the confidence estimator module to support the error detection on the existence (or nonexistence) of certain clauses.
- **Domain Knowledge Problem.** Other major failures are caused by the limited understanding of the underlying NL2SQL model over domain-specific semantics expressed in the NL queries. Since most of the existing NL2SQL models are based on pre-trained language models for semantic parsing work, it may not be possible to capture the semantics that is specific to certain databases. For example, following is an example from Spider validation set,

NL Query: *What is average, minimum, maximum age for all French singers?*
Predicted SQL of IKnow-SQL:

```
SELECT avg(age), min(age), max(age)
FROM singer WHERE is_male = 'French'
```

Gold SQL:

```
SELECT avg(age), min(age), max(age)
FROM singer WHERE country = 'France'
```

Since the model may fail to link the column "country" with the mentioned term "*French*" by token-level matching, the model mistakenly chooses the column "is_male" for the **WHERE** clause instead. To some extent, such failures can be mitigated by examining the data stored in databases for the schema linking process, but in the view of adapting to a new domain, providing some more external knowledge may be indispensable.

6 Conclusion

This paper presented IKnow-SQL, a novel approach designed to facilitate seamless access to underlying databases for end users of web applications. IKnow-SQL introduces a unified framework for any NL2SQL models to improve their translation accuracy through minimal interactions. IKnow-SQL first leverages the NL2SQL model to get a raw prediction for a given user query. IKnow-SQL then detects some potential errors in the prediction by measuring its confidence. Next, IKnow-SQL synthesizes clarification questions to solicit user feedback. Finally, IKnow-SQL memorizes the knowledge provided by the user for each interaction and outputs the final prediction until no low confidence is detected. Experimental results showed that IKnow-SQL can significantly improve the performance of existing state-of-the-art NL2SQL models with less user feedback.

References

1. Androutsopoulos, I., et al.: Natural language interfaces to databases - an introduction. Nat. Lang. Eng. **1**(1), 29–81 (1995)
2. Baik, C., et al.: Bridging the semantic gap with SQL query logs in natural language interfaces to databases. In: ICDE (2019)
3. Bogin, B., et al.: Representing schema structure with graph neural networks for text-to-SQL parsing. In: ACL (2019)
4. Cao, R., et al.: LGESQL: line graph enhanced text-to-SQL model with mixed local and non-local relations. In: ACL (2021)
5. Castaldo, N., Daniel, F., Matera, M., Zaccaria, V.: Conversational data exploration. In: Bakaev, M., Frasincar, F., Ko, I.-Y. (eds.) ICWE 2019. LNCS, vol. 11496, pp. 490–497. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19274-7_34
6. Chaurasia, S., et al.: Dialog for language to code. In: IJCNLP, pp. 175–180 (2017)
7. Desolda, G., et al.: Rapid prototyping of chatbots for data exploration. In: BCNC, pp. 5–10 (2021)

8. Fan, Y., et al.: Gar: a generate-and-rank approach for natural language to SQL translation. In: ICDE (2023)
9. Feng, L., Lu, H.: Integrating database and world wide web technologies. WWWJ **1**(2), 73–86 (1998)
10. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: ICML, vol. 48, pp. 1050–1059 (2016)
11. Goyal, K., et al.: Differentiable scheduled sampling for credit assignment. In: ACL, pp. 366–371 (2017)
12. Grave, E., et al.: Improving neural language models with a continuous cache. In: ICLR (2017)
13. Guo, J., et al.: Towards complex text-to-SQL in cross-domain database with intermediate representation. In: ACL (2019)
14. Gur, I., et al.: DialSQL: dialogue based structured query generation. In: ACL (2018)
15. He, H., et al.: Towards deeper understanding of the search interfaces of the deep web. WWWJ **2**, 133–155 (2007)
16. He, L., et al.: Human-in-the-loop parsing. In: EMNLP, pp. 2337–2342 (2016)
17. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: ICLR (2017)
18. Li, F., Jagadish, H.V.: Constructing an interactive natural language interface for relational databases. PVLDB **8**(1), 73–84 (2014)
19. Li, J., et al.: Graphix-T5: mixing pre-trained transformers with graph-aware layers for text-to-SQL parsing. In: AAAI, pp. 13076–13084 (2023)
20. Lin, X.V., et al.: Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In: EMNLP (2020)
21. Nakatsuji, M., et al.: Knowledge-aware response selection with semantics underlying multi-turn open-domain conversations. In: World Wide Web, pp. 1–16 (2023)
22. Niehues, J., et al.: Modeling confidence in sequence-to-sequence models. In: INLG, pp. 575–583 (2019)
23. OpenAI: GPT-4 technical report. CoRR (2023)
24. Pourreza, M., Rafiei, D.: DIN-SQL: decomposed in-context learning of text-to-SQL with self-correction. CoRR (2023)
25. Saha, D., et al.: ATHENA: an ontology-driven system for natural language querying over relational data stores. PVLDB **9**(12), 1209–1220 (2016)
26. Scholak, T., et al.: PICARD: parsing incrementally for constrained auto-regressive decoding from language models. In: EMNLP (2021)
27. Sen, J., et al.: ATHENA++: natural language querying for complex nested SQL queries. PVLDB **13**(11), 2747–2759 (2020)
28. Shi, P., et al.: Learning contextual representations for semantic parsing with generation-augmented pre-training. In: AAAI (2021)
29. Simitsis, A., et al.: Précis: from unstructured keywords as queries to structured databases as answers. PVLDB **17**(1), 117–149 (2008)
30. Su, Y., et al.: Natural language interfaces with fine-grained user interaction: a case study on web APIs. In: SIGIR, pp. 855–864 (2018)
31. Sukhbaatar, S., et al.: End-to-end memory networks. In: NeurIPS, pp. 2440–2448 (2015)
32. Touvron, H., et al.: LLaMA: open and efficient foundation language models. CoRR (2023)
33. Wang, B., et al.: RAT-SQL: relation-aware schema encoding and linking for text-to-SQL parsers. In: ACL (2020)

34. Xu, X., et al.: SQLNet: generating structured queries from natural language without reinforcement learning. CoRR (2017)
35. Yao, Z., et al.: Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. In: AAAI, pp. 2547–2554 (2019)
36. Yao, Z., et al.: Model-based interactive semantic parsing: a unified framework and a text-to-SQL case study. In: EMNLP, pp. 5446–5457 (2019)
37. Yu, T., et al.: TypeSQL: knowledge-based type-aware neural text-to-SQL generation. In: NAACL (2018)
38. Yu, T., et al.: CoSQL: a conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In: EMNLP (2019)
39. Zettlemoyer, L.S., Collins, M.: Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In: UAI (2005)
40. Zhang, R., et al.: Editing-based SQL query generation for cross-domain context-dependent questions. In: EMNLP, pp. 5337–5348 (2019)