An Imitation Game for Learning Semantic Parsers from User Interaction

Ziyu Yao¹, Yiqi Tang¹, Wen-tau Yih², Huan Sun¹, Yu Su¹
{o.470, tang.1466, sun.397, su.809}@osu.edu
scottyih@fb.com
¹The Ohio State University
²Facebook AI Research, Seattle

Abstract

Despite the widely successful applications, bootstrapping and fine-tuning semantic parsers are still a tedious process with challenges such as costly data annotation and privacy risks. In this paper, we suggest an alternative, human-in-the-loop methodology for learning semantic parsers directly from users. A semantic parser should be introspective of its uncertainties and prompt for user demonstration when uncertain. In doing so it also gets to imitate the user behavior and continue improving itself autonomously with the hope that eventually it may become as good as the user in interpreting their questions. To combat the sparsity of demonstration, we propose a novel annotation-efficient imitation learning algorithm, which iteratively collects new datasets by mixing demonstrated states and confident predictions and re-trains the semantic parser in a Dataset Aggregation fashion (Ross et al., 2011). We provide a theoretical analysis of its cost bound and also empirically demonstrate its promising performance on the text-to-SQL problem.

1 Introduction

Semantic parsing has found tremendous applications in building natural language interfaces that allow users to query data and invoke services without programming (Woods, 1973; Zettlemoyer and Collins, 2005; Berant et al., 2013; Su et al., 2017; Yu et al., 2018). The lifecycle of a semantic parser typically consists of two stages: (1) bootstrapping, where we keep collecting labeled data via trained annotators and/or crowdsourcing for model training until it reaches commercial-grade performance (e.g., 95% accuracy on a surrogate test set), and (2) fine-tuning, where we deploy the system, analyze the usage, and collect and annotate new data to address the identified problems or emerging needs.

However, it poses several challenges for scaling up or building semantic parsers for new domains: (1) high bootstrapping cost because mainstream neural parsing models are data-hungry and annotation cost of semantic parsing data is relatively high, (2) high fine-tuning cost from continuously analyzing usage and annotating new data, and (3) privacy risks arising from exposing private user conversations to annotators and developers (Lomas, 2019).

In this paper, we suggest an alternative methodology for building semantic parsers that could potentially address all the aforementioned problems. The key is to involve human users in the learning loop. A semantic parser should be introspective of its uncertainties and proactively prompt for demonstration from the user, who knows the question best, to resolve them. In doing so, the semantic parser

---

¹Code will be available at https://github.com/sunlab-osu/MISP.
would be able to accumulate targeted training data and continue improving itself autonomously without involving any annotators or developers, hence also minimizing privacy risks. The bootstrapping cost could also be significantly reduced because an interactive system needs not to be almost perfectly accurate to be deployed. On the other hand, such interaction opens up the black box and allows users to know more about the reasoning underneath the system and better interpret the final results (Su et al., 2018). A human-in-the-loop methodology like this also opens the door for domain adaptation and personalization.

This work builds on the recent line of research on interactive semantic parsing (Li and Jagadish, 2014; Chaurasia and Mooney, 2017; Gur et al., 2018; Yao et al., 2019b). Specifically, Yao et al. (2019b) provide a general framework, MISP (Model-based Interactive Semantic Parsing), which handles uncertainty modeling and natural language generation. We will leverage MISP for user interaction to prove the feasibility of the envisioned methodology. However, existing studies only focus on interacting with users to resolve uncertainties. None of them has answered the crucial question of how to learn from user interaction, which is the technical focus of this study.

One form of user interaction explored for learning semantic parsers is asking users to validate the execution results (Clarke et al., 2010; Iyer et al., 2017). While appealing, in practice it may be a difficult task for real users because they would not need to ask the question if they knew the answer in the first place. We instead aim to learn semantic parsers from fine-grained interaction where users only need to answer simple questions covered by their background knowledge (Figure 1). However, learning signals from such fine-grained interactions are bound to be sparse because the system needs to avoid asking too many questions and overwhelming the user, which poses a challenge for learning.

To this end, we propose a novel annotation-efficient imitation learning algorithm for learning semantic parsers from such sparse, fine-grained demonstration: The agent (semantic parser) only requests for demonstration when it is uncertain about a state (parsing step). For the certain/confident states, the actions chosen by the current policy are deemed as correct. The policy is updated iteratively in a Dataset Aggregation fashion (Ross et al., 2011). At each iteration, all the state-action pairs, demonstrated or confident, are included to form a new training set and train a new policy in a supervised way. Intuitively, using confident predictions for training mitigates the sparsity issue, but it may also introduce noise. We provide a theoretical analysis of the proposed algorithm and show that, under mild assumptions, the quality of the final policy is mainly determined by the quality of the initial policy and confidence estimation accuracy.

Using simulated users, we also empirically compare our method with a number of baselines on the text-to-SQL parsing problem, including the powerful but costly baseline of full expert annotation. On the WikiSQL (Zhong et al., 2017) dataset, compared with the full annotation baseline, we show that, when bootstrapped using only 10% of the training data, our method can achieve almost the same test accuracy (2% absolute loss) while using less than 10% of the annotations, without even taking into account the different unit cost of annotation from users vs. domain experts. We also show that the quality of the final policy is largely determined by the quality of the initial policy, further confirming the theoretical analysis. Finally, we demonstrate that the system can generalize to more complicated semantic parsing tasks such as Spider (Yu et al., 2018).

2 Related Work

Interactive Semantic Parsing. Our work extends interactive semantic parsing, a recent idea that leverages system-user interactions to improve semantic parsing on the fly (Li and Jagadish, 2014; He et al., 2016; Chaurasia and Mooney, 2017; Su et al., 2018; Gur et al., 2018; Yao et al., 2019a,b). As an example, Gur et al. (2018) built a neural model to identify and correct error spans in a generated SQL query via dialogues. Yao et al. (2019b) further generalized the interaction framework by formalizing a model-based intelligent agent called MISP. Our system leverages MISP to support interactivity but focuses on developing an algorithm for continually improving the base parser from end user interactions, which has not been accomplished by previous work.

Feedback-based Interactive Learning.

Learning interactively from user feedback has been studied for machine translation (Nguyen et al., 2017; Petrushkov et al., 2018; Kreutzer and Riezler, 2019) and other NLP tasks (Sokolov et al., 2016; Gao et al., 2018; Hancock et al.,
Most relevant to this work, Hancock et al. (2019) constructed a chatbot that learns to request feedback when the user is unsatisfied with the system response, and then further improves itself periodically from the satisfied responses and feedback responses. The work reaffirms the necessity of human-in-the-loop autonomous learning systems like ours.

In the field of semantic parsing, Clarke et al. (2010) and Iyer et al. (2017) learned semantic parsers from user validation on the query execution results. However, often times it may not very practical to expect end users able to validate answer correctness (e.g., consider validating an answer “103” for the question “how many students have a GPA higher than 3.5” from a massive table). Active learning is also leveraged to selectively obtain gold labels for semantic parsing and save human annotations (Duong et al., 2018; Ni et al., 2020). Our work is complementary to this line of research as we focus on learning interactively from end users (not “teachers”).

Imitation Learning. Traditional imitation learning algorithms (Daumé et al., 2009; Ross and Bagnell, 2010; Ross et al., 2011; Ross and Bagnell, 2014) iteratively execute and train a policy by collecting expert demonstrations for every policy decision. Despite its efficacy, the learning demands costly annotations from experts. In contrast, we save expert effort by selectively requesting for demonstrations. This idea is related to active imitation learning (Chernova and Veloso, 2009; Kim and Pineau, 2013; Judah et al., 2014; Zhang and Cho, 2017). For example, Judah et al. (2014) used active learning to select informative trajectories from the unlabeled data pool for expert demonstrations. However, their setting assumes a “teacher” to intentionally provide labels and an unlabeled data pool, while our algorithm targets at end users who are using the system. Similar to our approach, Chernova and Veloso (2009) solicited expert demonstrations only for uncertain states. However, their algorithm simply abandons policy actions that are confident, leading to sparse training data. Instead, our algorithm utilizes confident policy actions to combat the sparsity issue and is additionally provided with a theoretical analysis.

3 Preliminaries

Formally, we assume the semantic parsing model generates a semantic parse by executing a sequence of actions $a_t$ (parsing decisions) at each time step $t$. In practice, the definition of action depends on the specific semantic parsing model, as we will illustrate shortly. A state $s_t$ is then defined as a tuple of $(q, a_{1:t-1})$, where $q$ is the initial natural language question and $a_{1:t-1} = (a_1,...,a_{t-1})$ is the current partial parse. In particular, the initial state $s_1 = (q, \phi)$ contains only the question. Denote a semantic parser as $\hat{\pi}$, which is a policy function (Sutton and Barto, 2018) that takes a state $s_t$ as input and outputs a probability distribution over the action space. The semantic parsing process can be formulated as sampling a trajectory $\tau$ by alternately observing a state and sampling an action from the policy, i.e., $\tau = (s_1, a_1 \sim \hat{\pi}(s_1), ..., s_T, a_T \sim \hat{\pi}(s_T))$, assuming a trajectory length $T$. The probability of the generated semantic parse becomes:

$$p_{\hat{\pi}}(a_1:T|s_1) = \prod_{t=1}^{T} p_{\hat{\pi}}(a_t|s_t).$$

An interactive semantic parser typically follows the aforementioned definition and requests the user’s validation of a specific action $a_t$. Based on the feedback, a correct action $a_t^*$ can be inferred to replace the original $a_t$. The parsing process continues with $a_t^*$ afterwards. In this work, we adopt MISP (Yao et al., 2019b) as the back-end interactive semantic parsing framework, which enables system-user interaction via a policy probability based uncertainty estimator, a grammar-based natural language generator, and a multi-choice question-answering interaction design, as shown in Figure 1.

Example. Consider the SQLova parser (Hwang et al., 2019), which generates a query by filling “slots” in a pre-defined SQL sketch “SELECT $\text{Agg}$ $\text{SCol}$ WHERE $\text{WCol}$ OP $\text{VAL}$”. To complete the SQL query in Figure 1, it first takes three steps: $\text{WCol}=$“School/Club Team” ($a_1$), $\text{Agg}=$“COUNT” ($a_2$) and $\text{WCol}=$“School/Club Team” ($a_3$). MISP detects that $a_3$ is uncertain because its probability is lower than a pre-specified threshold. It validates $a_3$ with the user and corrects it with $\text{WCol}=$“Player” ($a_3^*$). The parsing continues with $\text{OP}=$“=” ($a_4$) and $\text{VAL}=$“jalen rose” ($a_5$). The trajectory length $T = 5$ in this case.

4 Learning Semantic Parsers from User Interaction

In this section, we present an imitation learning algorithm for learning semantic parsers from user
interactions. The algorithm is annotation-efficient and can train a parser without requiring a large amount of user feedback (or “annotations”), an important property for practical use in an end-user-facing system. Note that while we apply this algorithm to semantic parsing in this work, in principle the algorithm can be applied to other structured prediction tasks (e.g., text summarization or machine translation) as well.

4.1 An Imitation Learning Formulation

Under the interactive semantic parsing framework, a learning algorithm intuitively can aggregate \((s_t, a_t^*)\) pairs collected from user interactions and trains the parser to enforce \(a_t^*\) under the state \(s_t = (q, a_{1:t-1})\). However, this is not achievable by conventional supervised learning since the training needs to be conducted in an interactive environment, where the partial parse \(a_{1:t-1}\) is generated by the parser itself.

Instead, we formulate it as an imitation learning problem (Daumé et al., 2009; Ross and Bagnell, 2010; Ross et al., 2011; Ross and Bagnell, 2014). Consider the user as a demonstrator, then the derived action \(a_t^*\) can be viewed as an expert demonstration which is interactively sampled from the demonstrator’s policy (or expert policy) \(\pi^*\), i.e., \(a_t^* \sim \pi^*(s_t)\). The goal of our algorithm is thus to train policy \(\hat{\pi}\) to imitate the expert policy \(\pi^*\). A general procedure is described in Algorithm 1, where \(\hat{\pi}\) is learned iteratively for every \(m\) user questions (Line 1–9).

4.2 Annotation-efficient Imitation Learning

Consider parsing a user question and collecting training data using the parser \(\hat{\pi}_i\) in the \(i\)-th iteration (Line 5). A standard imitation learning algorithm such as DAGGER (Ross et al., 2011) usually requests expert demonstration \(a_t^*\) for every state \(s_t\) in the sampled trajectory. However, it requires a considerable amount of user annotations, which may not be practical when interacting with end users.

We propose an annotation-efficient imitation learning algorithm, which saves user annotations by selectively requesting user intervention, as shown in function PARSE&COLLECT. Specifically, in each parsing step, the system first previews whether it is confident about its own decision \(a_t\) (Line 13–14), which is determined when its probability is no less than a threshold, i.e., \(p_{\hat{\pi}_i}(a_t|s_t) \geq \mu\). In this case, the algorithm executes and collects the policy action \(a_t\) (Line 15–16); otherwise, a system-user interaction will be triggered and the derived demonstration \(a_t^* \sim \pi^*(s_t)\) will be collected and executed to continue parsing (Line 17–22).

Denote a collected state-action pair as \((s_t, \tilde{a}_t)\), where \(\tilde{a}_t\) could be \(a_t\) or \(a_t^*\) depending on whether an interaction is requested. To train \(\hat{\pi}_{i+1}\) (Line 7), our algorithm adopts a reduction-based approach similar to DAGGER and reduces imitation learning to iterative supervised learning. Formally, we define our training loss function as a weighted negative log-likelihood:

\[
\mathcal{L}(\hat{\pi}_{i+1}) = -\frac{1}{|D|} \sum_{(s_t, a_t, w_t) \in D} w_t \log p_{\hat{\pi}_i}(\tilde{a}_t|s_t),
\]

Algorithm 1 Learning from User Interaction

| Input: | Initial training data \(D_0\), confidence threshold \(\mu\). |
| Output: | A trained policy \(\hat{\pi}\). |

1: Initialize \(D \leftarrow D_0\).
2: Initialize \(\hat{\pi}_1\) by training it on \(D_0\).
3: for \(i = 1\) to \(N\) do
4: Observe \(m\) user questions \(q_j, j \in [1, m]\);
5: \(D_i \leftarrow \bigcup_{j=1}^m\) PARSE&COLLECT\((\mu, q_j, \hat{\pi}_i, \pi^*)\);
6: Aggregate dataset \(D \leftarrow D \cup D_i\);
7: Train policy \(\hat{\pi}_{i+1}\) on \(D\) using Eq. (1).
8: end for
9: return best \(\hat{\pi}_i\) on validation.

10: function PARSE&COLLECT\((\mu, q, \hat{\pi}_i, \pi^*)\)
11: Initialize \(D'_i \leftarrow \emptyset\), \(s_1 = (q, \phi)\).
12: for \(t = 1\) to \(T\) do
13: Preview action \(a_t = \arg\max_{a_t} \hat{\pi}_i(s_t)\);
14: if \(p_{\hat{\pi}_i}(a_t|s_t) \geq \mu\) then
15: \(w_t \leftarrow 1\);
16: Collect \(D'_i \leftarrow D'_i \cup \{(s_t, a_t, w_t)\}\);
17: Execute \(a_t\);
18: else
19: Trigger user interaction and derive expert demonstration \(a_t^* \sim \pi^*(s_t)\);
20: \(w_t \leftarrow 1\) if \(a_t^*\) is valid; 0 otherwise;
21: Collect \(D'_i \leftarrow D'_i \cup \{(s_t, a_t^*, w_t)\}\);
22: Execute \(a_t^*\);
23: end if
24: end for
25: return \(D'_i\).
26: end function
where $D$ is the aggregated training data over $i$ iterations and $w_i$ denotes the weight of $(s_i, a_i)$.

We consider assigning weight $w_i$ in three cases: (1) For confident actions $a_i$, we set $w_i = 1$. This essentially treats confident actions as gold decisions, which resembles self-training (Nigam and Ghani, 2000). (2) For user-confirmed decisions (valid demonstrations $a^*_j$), such as enforcing a WHERE condition on “Player” in Figure 1, $w_i$ is also set to 1 to encourage the parser to imitate the correct decisions from users. (3) For uncertain actions that cannot be addressed via human interactions (invalid demonstrations $a^*_j$, we assign $w_i = 0$. This could happen when some of the incorrect precedent actions are not fixed. For example, in Figure 1, if the system missed correcting the WHERE condition on “School/Club Team”, then whatever value it generates after “WHERE School/Club Team=” is wrong, and thus any action $a^*_j$ derived from human feedback would be invalid. A possible training strategy in this case can set $w_i$ to be negative, similar to (Welleck et al., 2020).

However, empirically we find this strategy fails to train the parser to correct its mistake in generating School/Club Team but rather disturbs model training. To solve this problem, we directly set $w_i = 0$ to remove the impact of unaddressed actions. A similar solution is also adopted in (Petrushkov et al., 2018; Kreutzer and Riezler, 2019). As shown in Section 6, this way of training weight assignment enables stable improvement in iterative model learning while requiring fewer user annotations.

5 Theoretical Analysis

While our system enjoys the benefit of learning from a small amount of user feedback, one crucial question is whether it can still achieve the same level of performance as a system trained on full expert annotations, if one could afford that and manage the privacy risk. In other words, what is the performance gap between our system and a fully supervised system? In this section, we answer this question by showing that the performance gap is mainly bounded by the learning policy’s probability of trusting a confident action that turns out to be wrong, which can be controlled in practice.

In the analysis, we follow prior work (Ross and Bagnell, 2010; Ross et al., 2011) to assume a unified trajectory length $T$ and focus the proof on the “infinite sample” case, which assumes an infinite number of samples in each iteration (i.e., $m = \infty$ in Algorithm 1), such that the state space can be fully explored by the current policy. An analysis under the “finite sample” case can be found in Appendix A.5.

5.1 Cost Function for Analysis

Different from typical imitation learning tasks (e.g., Super Tux Kart (Ross et al., 2011)), in semantic parsing, there exists only one gold trajectory semantically identical to the question and can return correct execution results.\(^3\) Whenever a policy action is different from the gold one, the whole trajectory will not yield the correct semantic meaning. Therefore, we analyze a policy’s performance only when it is conditioned on a gold partial parse, i.e., $s_t \in d^t_{\pi^*}$, where $d^t_{\pi^*}$ is the state distribution in step $t$ when executing the expert policy $\pi^*$ for first $t$-1 steps. Let $\ell(s, \hat{\pi}) = 1 - p\alpha(\alpha = a^*|s)$ be the loss of $\hat{\pi}$ making a mistake at state $s$. By summing up a policy’s expected loss over $T$ steps, we define the cost of the policy as:

$$J(\hat{\pi}) = \sum_{t=1}^{T} \mathbb{E}_{s_t \in d^t_{\pi}} [\ell(s_t, \hat{\pi})]$$

$$= T \mathbb{E}_{s \sim d_{\pi^*}} [\ell(s, \hat{\pi})],$$

(2)

where $d_{\pi^*} = \frac{1}{T} \sum_{t=1}^{T} d^t_{\pi^*}$ denotes the average expert state distribution (assuming time step $t$ is a random variable uniformly sampled from $1 \sim T$). A detailed derivation is shown in Appendix A.1.

The better $\hat{\pi}$ is, the smaller this cost becomes. Although it is not exactly the same as the objective evaluated in experiments, which measures the correctness of a complete trajectory (rather than a single policy action) sampled from $\hat{\pi}$, this simplified version makes theoretical analysis easier and reflects a consistent relative performance among algorithms. Next, we will derive the bound of each policy’s cost in order to compare their performance.

5.2 Cost Bound of Supervised Approach

A fully supervised system trains a parser on expert-annotated $(q, a^*_{1:T})$ pairs, where the gold semantic parse $a^*_{1:T}$ can be viewed as generated by executing the expert policy $\pi^*$. This gives the policy $\pi_{sup}$:

$$\pi_{sup} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}} [l(s, \pi)],$$

\(^3\)We assume a canonical order for swappable components in a parse. In practice, it may be possible, though rare, for one question to have multiple gold parses.
where Π is the policy space induced by the model architecture. A detailed derivation in Appendix A.2 shows the cost bound of the supervised approach:

**Theorem 5.1.** For supervised approach, let $\epsilon_N = \min_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}}[\ell(s, \pi)]$, then $J(\hat{\pi}_{\text{sup}}) = T\epsilon_N$.

The theorem gives an exact bound (as shown by the equality) since the supervised approach, given the “infinite sample” assumption, trains a policy under the same state distribution $d_{\pi^*}$ as the one being evaluated in the cost function (Eq. (2)). As we will show next, when adopting an annotation-efficient learning strategy, our proposed algorithm breaks this consistency and thus induces a performance gap compared with the supervised approach.

### 5.3 Cost Bound of Our Proposed Algorithm

During its iterative learning, Algorithm 1 produces a sequence of policies $\hat{\pi}_{1:N} = (\hat{\pi}_1, \hat{\pi}_2, ..., \hat{\pi}_N)$, where $N$ is the number of training iterations, and returns the one with the best test-time performance on validation as $\hat{\pi}$ (Line 9). Recall that our algorithm samples a trajectory by executing actions from both the previously learned policy $\hat{\pi}_i$ and the expert policy $\pi^*$ (when an interaction is requested). Let $\pi_i$ denote such a “mixture” policy. The cost of the learned policy $\hat{\pi}$ can be bounded as:

$$J(\hat{\pi}) = \min_{\hat{\pi} \in \hat{\pi}_{1:N}} T\mathbb{E}_{s \sim d_{\pi^*}}[\ell(s, \hat{\pi})]$$

$$\leq T\frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi^*}}[\ell(s, \hat{\pi}_i)]$$

$$\leq T\frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi_i}}[\ell(s, \hat{\pi}_i)] + \ell_{max}||d_{\pi_i} - d_{\pi^*}||_1$$

The above derivation shows that the bound comprises of two parts. The first term $\mathbb{E}_{s \sim d_{\pi_i}}[\ell(s, \hat{\pi}_i)]$ calculates the expected training loss of $\hat{\pi}_i$. Notice that, in training, each trajectory is sampled from the mixture policy ($s \sim d_{\pi_i}$), while in evaluation, we measure a policy’s performance conditioned on a gold partial parse ($s \sim d_{\pi^*}$ in Eq. (2)). This discrepancy, which does not exist in the supervised approach, explains the performance loss of our algorithm, which is bounded by the second term $\ell_{max}||d_{\pi_i} - d_{\pi^*}||_1$, the $L_1$ distance between $d_{\pi_i}$ and $d_{\pi^*}$ weighted by the maximum loss value $l_{max}$ that $\hat{\pi}_i$ encounters over the training. Bounding the two terms gives the following theorem:

**Theorem 5.2.** For the proposed annotation-efficient imitation learning algorithm, if $N$ is $O(T)$, there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $J(\hat{\pi}) \leq T[\epsilon_N + \frac{2\ell_{max}}{N} \sum_{i=1}^{N} \epsilon_i] + O(1)$.

Here, $\epsilon_N = \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi_i}}[\ell(s, \pi)]$ denotes the best expected policy loss in hindsight, and $\epsilon_i$ denotes the probability that $\hat{\pi}_i$ does not query the expert policy (i.e., being confident) but its own action is wrong under $d_{\pi^*}$. A detailed derivation can be found in Appendix A.3–A.4.

**Remarks.** A comparison of Theorem 5.1 and Theorem 5.2 shows that the performance gap led by our algorithm is mainly bounded by $\frac{1}{N} \sum_{i=1}^{N} \epsilon_i$. Intuitively this is because whenever a learning policy in our algorithm collects its own, but wrong, action as the gold one for training, it introduces noise that does not exist in the supervised approach’s training set. This finding inspires us to restrict the gap by lowering down the learning policy’s error rate when it does not query the expert. Empirically this can be achieved by setting:

- **Accurate policy confidence estimation**, such that actions regarded confident are generally correct.
- **Moderate model initialization**, such that generally the policy is less likely to make wrong actions throughout the iterative training.

For the first point, we set a high confidence threshold $\mu$, which has been demonstrated to be reliable for MISP (Yao et al., 2019b). In the future, it can even be replaced by a machine learning module (see a discussion in Section 7). We empirically validate the second point in our experiments.

### 6 Experiments

In this section, we conduct experiments to demonstrate the annotation efficiency of our algorithm (Section 4) and that it can train semantic parsers to reach high performance when the system is reasonably instantiated, consistent with our theoretical analysis in Section 5.

#### 6.1 Experimental Setup

We test our system on the WikiSQL dataset (Zhong et al., 2017). The dataset contains a large scale of annotated question-SQL pairs (56,355 pairs for training) and thus serves as a good resource for experimenting iterative learning. For the base semantic parser, we choose SQLova (Hwang et al., 2019), one of the top-performing models on WikiSQL, to ensure a reasonable model capacity in terms of data utility along iterative training.
To instantiate the proposed algorithm, we set a high confidence threshold $\mu = 0.95$ following Yao et al. (2019b) and experiment with different initialization settings as suggested by our analysis in Section 5, using 10%, 5% and 1% of the total training data. During iterative learning, questions from the remaining training data arrive in a random order to simulate user questions. The parser is trained with simulated user feedback (which is obtained by directly comparing the synthesized query with the gold one) iteratively for every $m = 1,000$ questions. We test systems under different training iterations $N$ and report results averaged over three random runs. More implementation details are included in Appendix B.1.

6.2 System Comparison

We compare our system (denoted as MISP-L since it builds a Learning algorithm upon MISP) with the traditional supervised approach (denoted as Full Expert). To investigate the skyline capability of our system, we also present a variant called MISP-L*, which is assumed with perfect confidence measurement and interaction design, so that it can precisely identify and correct its mistakes during parsing. This is implemented by allowing the system to compare its synthesized query with the gold one. Note that this is not a realized automatic system; we show its performance as an upper bound of MISP-L.

On the other hand, while the learning systems by Clarke et al. (2010) and Iyer et al. (2017), which request user validation on query execution results, may not very practical to interact with end users, we include them nonetheless in the interest of comprehensive comparison. This leads to two baseline systems. The Binary User system requests binary user feedback on whether executing the generated SQL query returns correct database results and collects only queries with correct execution results to further improve the parser, similar to (Clarke et al., 2010). The Binary User+Expert system additionally collects full expert SQL annotations when the execution results of the generated SQL queries are wrong, similar to (Iyer et al., 2017).

6.3 Experimental Results

We evaluate each system by answering the two research questions (RQs):

- **RQ1:** Can the system learn a semantic parser without requiring a large amount of annotations?

- **RQ2:** For interactive systems, while requiring weaker supervision, can they train the parser to reach a performance comparable to the traditional supervised system?

For RQ1, we measure the number of user/expert annotations a system requires to train a parser. For Full Expert, this number is equal to the trajectory length of the gold query (e.g., 5 for the query in Figure 1); for MISP-L and MISP-L*, it is the number of user interactions during training. For Binary User(+Expert), it is hard to quantify “one annotation”, which varies according to the actual database size and the query difficulty. In experiments, we approximate this number by calculating it in the same way as Full Expert, with the assumption that in general validating an answer is as hard as validating the SQL query itself. More accurate metrics can be explored by conducting a user study, as we discussed in Section 7. Note that while we do not differentiate the actual cost (e.g., time and financial cost) of users and experts in this aspect, we emphasize that our system enjoys an additional benefit of collecting training examples from a much cheaper and more abundant source while serving end users’ needs at the same time.

Figure 2 (top) shows each system’s parsing accuracy on WikiSQL test set after they have been trained on certain amounts of annotations. Consistently under all initialization conditions, MISP-L consumes a comparable or smaller amount of annotations to train the parser to reach the same parsing accuracy. As shown in Figure 5 in Appendix, MISP-L requires an average of no more than one interaction for most questions along the iterative training. Given the limited size of WikiSQL training set, the simulation experiments currently can only show the system’s performance under a small number of annotations. However, we expect this gain to continue as it receives more user questions in the long-term deployment.

To answer RQ2, Figure 2 (bottom) compares each system’s parsing accuracy after they have been trained for the same number of iterations. The results demonstrate that when a semantic parser is moderately initialized (10%/5% initialization setting), MISP-L can further improve it to reach a comparable accuracy as Full Expert (0.776/0.761 vs. 0.794 in the last iteration). In the extremely weak 1% initialization setting (using only around 500 initial training examples), all interactive learning systems suffer from a huge performance loss.
This is consistent with our finding in theoretical analysis (Section 5). In Appendix C, we plot the value of $e_i$, the probability that $\hat{\pi}_i$ makes a confident but wrong decision given a gold partial parse, showing that a better initialized policy generally obtains a smaller $e_i$ throughout the training and thus a tighter cost bound.

For both RQ1 and RQ2, our system surpasses Binary User, the execution feedback-based system. In experiments, we find out that the inferior performance of Binary User is mainly due to the “spurious program” issue (Guu et al., 2017), i.e., a SQL query having correct execution results can still be incorrect in terms of semantics. MISP-L circumvents this issue by directly validating the semantic meaning of intermediate parsing decisions.

Finally, when it is assumed with perfect interaction design and confidence estimator, MISP-L* shows striking superiority in both aspects. Since it always corrects wrong decisions immediately, MISP-L* can collect and derive the same training examples as Full Expert, and thus trains the parser to Full Expert’s performance level. Meanwhile, it requires only 10% of the annotations that Full Expert consumes. These observations implies large room for MISP-L to be improved in the future.

6.4 Generalize to Complex SQL Queries
Since queries in WikiSQL are generally simple and follow a pre-specified “SELECT...WHERE...” sketch, the last part of our experiments investigates whether our system can generalize to complex SQL queries. To this end, we test our system on the Spider dataset (Yu et al., 2018), where SQL queries can contain complicated keywords like GROUP BY. For the base semantic parser, we choose EditSQL (Zhang et al., 2019), one of the open-sourced top models on Spider. Given the small size of Spider (7,377 question-SQL query pairs for training after data cleaning), we only experiment with one initialization setting, using 10% of the training set. Since all Spider models do not predict the specific values in a SQL query (e.g., “jalen rose” in Figure 1), we cannot execute the generated query to simulate the binary execution feedback. Therefore, we only compare our system with the Full Expert baseline. Parsers are evaluated on Spider Dev set since the test set is not publicly available. We include all implementation details in Appendix B.2.

Figure 3 (top) shows that our system and its variant consistently achieve comparable or better performance.
7 Conclusion and Future Work

In this paper, we explore building an interactive semantic parser that continually improves itself from end user interaction, without involving annotators or developers. To this end, we propose an annotation-efficient imitation learning algorithm to learn from the sparse, fine-grained demonstrations. We prove the quality of the algorithm theoretically and show its advantage over the traditional full expert annotation approach via experiments.

As a pilot study on this research topic, we train systems with simulated user feedback. One important future work is to conduct a large-scale user study and collect interactions from real users. This is not trivial and has to account for uncertainties such as noisy user feedback. By analyzing real users’ statistics (e.g., average time spent on each question), we believe a more accurate and realistic formulation of user/expert annotation cost can be derived to guide future research.

Besides, we would like to explore more accurate confidence measurement to improve our system, as suggested by our theoretical analysis. In experiments, we observe that the two neural semantic parsers (especially the more complicated EditSQL) tend to be overconfident, and training them with more data does not mitigate this issue. To address that, future directions include neural network calibration (Guo et al., 2017) and using machine learning components (e.g., a reinforcement learning-based active selector (Fang et al., 2017)) to replace the confidence threshold.

Finally, the proposed annotation-efficient imitation learning algorithm can be generalized to other NLP tasks (Sokolov et al., 2016) and classical imitation learning problems (Ross et al., 2011). We expect this algorithm to save human annotation effort particularly for low-resource tasks (Mayhew et al., 2019).

Acknowledgments

This research was sponsored in part by the Army Research Office under cooperative agreements W911NF-17-1-0412, NSF Grant IIS1815674, Fujitsu gift grant, and Ohio Supercomputer Center (Center, 1987). The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Kazuoki Azuma. 1967. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Ohio Supercomputer Center. 1987. Ohio supercomputer center. [http://osc.edu/ark:/19495/f5s1ph73](http://osc.edu/ark:/19495/f5s1ph73).

Shobhit Chaurasia and Raymond J Mooney. 2017. Dialog for language to code. In *Proceedings of the Eighth International Joint Conference on Natural*
Sonia Chernova and Manuela Veloso. 2009. Interactive policy learning through confidence-based autonomy. Journal of Artificial Intelligence Research, 34:1–25.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world’s response. In Proceedings of the fourteenth conference on computational natural language learning, pages 18–27. Association for Computational Linguistics.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. Machine learning, 75(3):297–325.

Yang Gao, Christian M Meyer, and Iryna Gurevych. 2018. APRIL: Interactively learning to summarise by combining active preference learning and reinforcement learning. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4120–4130.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1321–1330. JMLR. org.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. DialSQL: Dialogue based structured query generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1339–1349.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1051–1062.

Elad Hazan, Amit Agarwal, and Satyen Kale. 2007. Logarithmic regret algorithms for online convex optimization. Machine Learning, 69(2-3):169–192.
Ansong Ni, Pengcheng Yin, and Graham Neubig. 2020. Merging weak and active supervision for semantic parsing. In Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI), New York, USA.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In Proceedings of the ninth international conference on Information and knowledge management, pages 86–93.

Pavel Petrushkov, Shahram Khadivi, and Evgeny Matysov. 2018. Learning from chunk-based feedback in neural machine translation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 326–331.

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 661–668.

Stéphane Ross and J. Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. ArXiv, abs/1406.5979.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 627–635.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016. Learning structured predictors from bandit feedback for interactive nlp. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1610–1620.

Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, Michael Gamon, and Mark Encarnacion. 2017. Building natural language interfaces to web apis. In Proceedings of the International Conference on Information and Knowledge Management.

Yu Su, Ahmed Hassan Awadallah, Miaozen Wang, and Ryen W White. 2018. Natural language interfaces with fine-grained user interaction: A case study on web APIs. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.

Richard S Sutton and Andrew G Barto. 2018. Reinforcement learning: An introduction. MIT press.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In International Conference on Learning Representations.

William A Woods. 1973. Progress in natural language understanding: an application to lunar geology. In Proceedings of the American Federation of Information Processing Societies Conference.

Ziyu Yao, Xiujun Li, Jianfeng Gao, Brian Sadler, and Huan Sun. 2019a. Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 2547–2554.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019b. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. 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 5450–5461.

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

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence, pages 658–666.

Jiakai Zhang and Kyunghyun Cho. 2017. Query-efficient imitation learning for end-to-end simulated driving. In Thirty-First AAAI Conference on Artificial Intelligence.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. 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 5341–5352.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103.
A Theoretical Analysis in Infinite Sample Case

In this section, we give a detailed theoretical analysis to derive the cost bound of the supervised approach and our proposed annotation-efficient imitation learning algorithm in Section 4.

Following Ross et al. (2011), we first focus the proof on an infinite sample case, which assumes an infinite number of samples to train a policy in each iteration (i.e., \( m = \infty \) in Algorithm 1). As an overview, we start the analysis by introducing the “cost function” we use to analyze each policy in Appendix A.1, which represents an inverse quality of a policy. In Appendix A.2, we derive the bound of the cost of the supervised approach. Appendix A.3 and Appendix A.4 then discuss the cost bound of our proposed algorithm. Finally, in Appendix A.5, we show the cost bound of our algorithm in finite sample case.

A.1 Cost Function for Analysis

In a semantic parsing task, whenever a policy action is different from the gold one, the whole trajectory cannot yield the correct semantic meaning. Therefore, we analyze a policy’s performance only when it is conditioned on a gold partial parse, i.e., \( \langle q, a_{1:t-1} \rangle \). By summing up the above expected cost over the \( T \) steps, we define the total cost of executing policy \( \pi \) for \( T \) steps as:

\[
J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{q \in Q}[C_{\pi}(q)]
\]

\[
= \sum_{t=1}^{T} \mathbb{E}_{\alpha_{t} \sim d_{\pi}^{\ast}}[1 - p(\alpha_{t} = a_{t}^{\ast} | s_{t})].
\]

Denote \( \ell(s, \pi) = 1 - p(\alpha = a^{\star} | s), a \sim \pi(s), a^{\star} \sim \pi^{*}(s) \) as the “loss function” in our analysis, which is bounded within \([0, 1]\), then the cost of policy \( \pi \) can be simplified as:

\[
J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{s_{t} \sim d_{\pi}^{\ast}}[\ell(s_{t}, \pi)]
\]

\[
= T \mathbb{E}_{s \sim \mathcal{U}(1, T)} \mathbb{E}_{s_{t} \sim d_{\pi}^{\ast}}[\ell(s_{t}, \pi)]
\]

\[
= T \mathbb{E}_{s \sim d_{\pi}^{\ast}}[\ell(s, \pi)], \tag{3}
\]

where \( d_{\pi}^{\ast} = \frac{1}{T} \sum_{t=1}^{T} d_{\pi}^{t} \) is the average expert state distribution, when we assume the time step \( t \) to be a random variable under the uniform distribution \( \mathcal{U}(1, T) \) (the second equality).

The better a policy \( \pi \) is, the smaller this cost becomes. Our analysis thus compares each policy by deriving the “bound” of their costs.

A.2 Derivation of Cost Bound for Supervised Approach

In this section, we analyze the cost bound for the supervised approach. Recall that the supervised approach trains a policy \( \tilde{\pi} \) using the standard supervised learning algorithm with supervision from \( \pi^{*} \) at every decision step. Therefore, it finds the best policy \( \tilde{\pi}_{\text{sup}} \) on infinite samples as:

\[
\tilde{\pi}_{\text{sup}} = \arg \min_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi}^{\ast}}[\ell(s, \pi)], \tag{4}
\]

where \( \Pi \) denotes the policy space induced by the model architecture, and the expectation over \( s \) is sampled from the whole \( d_{\pi}^{\ast} \) space because of the “infinite sample” assumption. The supervised approach thus obtains the following cost bound:

\[
J(\tilde{\pi}_{\text{sup}}) = T \mathbb{E}_{s \sim d_{\pi}^{\ast}}[\ell(s, \tilde{\pi}_{\text{sup}})]
\]

\[
= T \min_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi}^{\ast}}[\ell(s, \pi)].
\]

This gives the following theorem:
**Theorem A.1.** For supervised approach, let \( \epsilon_N = \min_{\pi \in \cal{P}} \mathbb{E}_{s \sim d_{\pi^*}}[\ell(s, \pi)] \), then \( J(\overline{\pi}_{sup}) = T \epsilon_N \).

The cost bound of the supervised approach represents its exact performance as implied by the equality. This is because the approach trains a policy (Eq. (4)) under the same state distribution \( d_{\pi^*} \) (given the “infinite sample” assumption) as in evaluation (Eq. (3)). As we will show next, the proposed annotation-efficient imitation learning algorithm breaks this consistency while enjoying the benefit of high annotation efficiency, which explains the performance gap.

**A.3 No-regret Assumption**

The derivation of our proposed annotation-efficient imitation learning algorithm’s cost bound leverages a “no-regret” assumption:

**Assumption A.1.** No-regret assumption. Define \( \ell_i(\pi) = \mathbb{E}_{s \sim d_{\pi}}[\ell(s, \pi)] \) and \( \epsilon_N = \min_{\pi \in \cal{P}} \frac{1}{N} \sum_{i=1}^{N} \ell_i(\pi) \), then

\[
\frac{1}{N} \sum_{i=1}^{N} \ell_i(\overline{\pi}_i) - \epsilon_N \leq \gamma_N
\]

for \( \lim_{N \to \infty} \gamma_N = 0 \) (usually \( \gamma_N \in \tilde{O}(\frac{1}{N}) \)).

Many no-regret algorithms (Hazan et al., 2007; Kakade and Tewari, 2009) that guarantee \( \gamma_N \in \tilde{O}(\frac{1}{N}) \) require convexity or strongly-convexity of the loss function. However, the loss function used in our application, which is built on the top of a deep neural network model, does not satisfy this requirement. In this analysis, we simplify the setting and directly make this assumption for convenience of the proof. A more accurate regret bound for non-convex neural networks can be researched in the future.

Another concern is that the collected online training labels come from not only the expert policy \( \pi^* \) (when it is queried), but also the learning policy \( \overline{\pi}_i \) (when the agent has a high confidence on its policy action). Labels from the learning policy may bring noise amid the model fitting to the expert policy. However, in practice the impact from such noisy labels are limited when the confidence threshold \( \mu \) is set at a high value (e.g., 0.95). In this case, labels from \( \overline{\pi}_i \) are generally clean and lead to increasing performance during iterative training. Therefore, it is still safe to make this no-regret assumption.

**A.4 Derivation of Cost Bound for Our Proposed Algorithm**

As shown in Algorithm 1, our algorithm produces a sequence of policies \( \overline{\pi}_{1:N} = (\overline{\pi}_1, \overline{\pi}_2, ..., \overline{\pi}_N) \), where \( N \) is the number of training iterations, and the algorithm returns the one with the best test-time performance on validation as \( \overline{\pi} \). In training, our algorithm executes actions from both the learning policy \( \overline{\pi}_i \) (when the model is confident) and the expert policy \( \pi^* \). We denote this “mixture” policy as \( \pi_i \). Then for the first \( N \) iterations, we have the cost bound of our algorithm as:

\[
J(\overline{\pi}) = \min_{\overline{\pi} \in \overline{\pi}_{1:N}} T \mathbb{E}_{s \sim d_{\pi^*}}[\ell(s, \overline{\pi})] \\
\leq \frac{T}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi_i}}[\ell(s, \overline{\pi}_i)] \\
\leq \frac{T}{N} \sum_{i=1}^{N} \left[ \mathbb{E}_{s \sim d_{\pi_i}}[\ell(s, \overline{\pi}_i)] + \ell_{max} ||d_{\pi_i} - d_{\pi^*}||_1 \right].
\]

(5)

From the last inequality, we can see that the cost bound of our algorithm is restricted by two terms. The first term \( \mathbb{E}_{s \sim d_{\pi_i}}[\ell(s, \overline{\pi}_i)] \) denotes the expected loss of \( \overline{\pi}_i \) under the state induced by \( \pi_i \) during training (under the “infinite sample” assumption, as mentioned in the beginning of the analysis). By applying the no-regret assumption (Assumption A.1), this term can be bound by \( \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi_i}}[\ell(s, \overline{\pi}_i)] \leq \epsilon_N + \gamma_N \). Here, \( \epsilon_N = \min_{\pi \in \cal{P}} \frac{1}{N} \sum_{i=1}^{N} \ell_i(\pi) \) denotes the best expected training loss in hindsight.

The second term denotes the \( L_1 \) distance between state distributions induced by \( \pi_i \) and \( \pi_i^* \), weighted by the maximum loss value \( l_{max} \) that \( \overline{\pi}_i \) encounters over the training. As we notice, unlike the supervised approach, our algorithm trains a policy under \( d_{\pi^*} \), which is different from the state distribution \( d_{\pi^*} \) used to evaluate the policy (Eq. (3)). This discrepancy explains the performance loss of our algorithm compared to the supervised approach and is bounded by the aforementioned \( L_1 \) distance. To further bound this term, we define \( e_i \) as the probability that \( \overline{\pi}_i \) makes a confident (i.e., without querying the expert policy) but wrong action under \( d_{\pi^*} \), and introduce the following lemma:

**Lemma A.1.** \( ||d_{\pi_i} - d_{\pi^*}||_1 \leq 2T e_i \).

**Proof.** Let \( \beta_{ij} \) be the probability of querying the expert policy under \( d_{\pi^*} \), \( \epsilon_{it} \) the error rate of \( \overline{\pi}_i \)
under $d_{\pi^*}$, and $d$ any state distribution besides $d_{\pi^*}$. We can then express $d_{\pi_i}$ by:

$$d_{\pi_i} = \prod_{t=1}^{T} (\beta_{it} + (1 - \beta_{it})(1 - \hat{e}_{it}))d_{\pi^*} + (1 - \prod_{t=1}^{T} (\beta_{it} + (1 - \beta_{it})(1 - \hat{e}_{it}))d.$$ 

The distance between $d_{\pi_i}$ and $d_{\pi^*}$ thus becomes

$$||d_{\pi_i} - d_{\pi^*}||_1 = (1 - \prod_{t=1}^{T} (\beta_{it} + (1 - \beta_{it})(1 - \hat{e}_{it}))) ||d - d_{\pi^*}||_1 \leq 2(1 - \prod_{t=1}^{T} (\beta_{it} + (1 - \beta_{it})(1 - \hat{e}_{it})))$$

$$\leq 2 \sum_{t=1}^{T} [1 - (\beta_{it} + (1 - \beta_{it})(1 - \hat{e}_{it}))]$$

$$\leq 2 \sum_{t=1}^{T} \hat{e}_{it}(1 - \beta_{it})$$

$$\leq 2 \sum_{t=1}^{T} e_{it}$$

$$= 2Te_i.$$ 

The second inequality uses $1 - \prod_{t=1}^{T} x_t \leq \sum_{t=1}^{T} (1 - x_t)$, which holds when $x_t \in [0, 1]$.

By applying Assumption A.1 and Lemma A.1 to Eq. (3), we derive the following inequality:

$$J(\hat{\pi}) \leq T[\gamma_N + \epsilon_N + \frac{2T\ell_{\max}}{N} \sum_{i=1}^{N} e_i].$$

Given a large enough $N$ ($N \in \tilde{O}(T)$), by the no-regret assumption, we can further simplify the above as:

$$J(\hat{\pi}) \leq T[\epsilon_N + \frac{2T\ell_{\max}}{N} \sum_{i=1}^{N} e_i] + O(1),$$

which leads to our theorem:

**Theorem A.2.** For our proposed annotation-efficient imitation learning algorithm, if $N$ is $\tilde{O}(T)$, there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $J(\hat{\pi}) \leq T[\epsilon_N + \frac{2T\ell_{\max}}{N} \sum_{i=1}^{N} e_i] + O(1).$

In experiments, we consider a skyline instantiation of the proposed algorithm, called MISP-L*.

This instantiation is assumed with perfect confidence estimation and interaction design, such that it can precisely detects and corrects its intermediate mistakes during parsing. Therefore, MISP-L* presents an upper bound performance (i.e., the tightest cost bound) of our algorithm. This can be interpreted theoretically. In fact, for MISP-L*, $e_i$ is always zero since the system has ensured that its policy action is correct when it does not query the expert policy. In this case, $d_{\pi_i} = d_{\pi^*}$, so $\epsilon_N = \min_{\pi \in \Pi} \sum_{i=1}^{N} E_{s \sim d_{\pi^*}}[\ell(s, \pi)] = \min_{\pi \in \Pi} E_{s \sim d_{\pi^*}}[\ell(s, \pi)].$ Therefore, according to Theorem A.2, MISP-L* has a cost bound of:

$$J(\hat{\pi}) \leq T\epsilon_N + O(1),$$

where $\epsilon_N = \min_{\pi \in \Pi} E_{s \sim d_{\pi^*}}[\ell(s, \pi)].$

By comparing this bound with the cost bound in Theorem A.1, it is observed that MISP-L* shares the same cost bound as the supervised approach (except for the inequality relation and the constant). This is explainable since MISP-L* indeed collects exactly the same training labels (from $\pi^*$) as the supervised approach.

**A.5 Cost Bound of Our Proposed Algorithm in Finite Sample Case**

The theorems in previous sections hold when the algorithm observes infinite trajectories. However, in practice, our algorithm will observe the training loss from only a finite set of $m$ trajectories at each iteration $i$ using $\pi_i$. For this consideration, in the following discussion, we provide a proof of the cost bound of our proposed algorithm under the finite sample case.

In the finite sample setting, our algorithm observes the training loss from a finite number of trajectories. We define $D_i$ as the $m$ trajectories collected in the $i^{th}$ iteration. In every iteration, the algorithm observes loss $\ell_i(\hat{\pi}_i) = E_{s \sim D_i}[\ell(s, \hat{\pi}_i)].$ By the no-regret assumption (Assumption A.1), the average observed loss for each iterations can still be bounded by the following inequality: $\frac{1}{N} \sum_{i=1}^{N} E_{s \sim D_i}[\ell(s, \pi_i)] \leq \tilde{\gamma}_N.$ We use $\tilde{\gamma}_N = \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} E_{s \sim D_i}[\ell(s, \pi)]$ to denote the loss of the best policy on the finite samples.

Following Eq. (5), we need to switch the derivation from the expected loss of $\hat{\pi}_i$ over $d_{\pi_i}$ (i.e., $E_{s \sim d_{\pi_i}}[\ell(s, \hat{\pi}_i)]$) to that over $D_i$ (i.e., $E_{s \sim D_i}[\ell(s, \hat{\pi}_i)]$), the actual state distribution that $\hat{\pi}_i$ is trained on. To fill this gap, we introduce $Y_{ij}$
to denote the difference between the expected loss of $\hat{\pi}_i$ under $d_{\pi_i}$ and the average loss of $\hat{\pi}_i$ under the $j^{th}$ sample trajectory with $\pi$ at iteration $i$. The random variables $Y_{ij}$ over all $i \in \{1, 2, ..., N\}$ and $j \in \{1, 2, ..., m\}$ are all zero mean, bounded in $[-\ell_{\text{max}}, \ell_{\text{max}}]$ and form a martingale in the order of $Y_{11}, Y_{12}, ..., Y_{1m}, Y_{21}, ..., Y_{NM}$. By Azuma-Hoeffding’s inequality (Azuma, 1967; Hoeffding, 1994), $\frac{1}{mN} \sum_{i=1}^{N} \sum_{j=1}^{m} Y_{ij} \leq \ell_{\text{max}} \sqrt{\frac{2 \log(1/\delta)}{mN}}$ with probability $1 - \delta$. Following the derivations in Eq. (5) and by introducing $Y_{ij}$, with probability of $1 - \delta$, we obtain the following inequalities by definition:

$$\begin{align*}
J(\hat{\pi}) & \leq \frac{T}{N} \sum_{i=1}^{N} \left[ \mathbb{E}_{s \sim d_s}[\ell(s, \hat{\pi}_i)] + \ell_{\text{max}} ||d_{\hat{\pi}_i} - d_{\pi^*}||_1 \right] \\
\leq \frac{T}{N} \sum_{i=1}^{N} \left[ \mathbb{E}_{s \sim D_i}[\ell(s, \hat{\pi}_i)] + \ell_{\text{max}} ||d_{\hat{\pi}_i} - d_{\pi^*}||_1 \right] \\
& \quad + \frac{T}{mN} \sum_{i=1}^{N} \sum_{j=1}^{m} Y_{ij} \\
\leq \frac{T}{N} \sum_{i=1}^{N} \left[ \mathbb{E}_{s \sim D_i}[\ell(s, \hat{\pi}_i)] + \ell_{\text{max}} ||d_{\hat{\pi}_i} - d_{\pi^*}||_1 \right] \\
& \quad + \ell_{\text{max}} T \sqrt{\frac{2 \log(1/\delta)}{mN}} \\
& \quad \leq T \left[ \tilde{\gamma}_N + \tilde{\epsilon}_N + \ell_{\text{max}} \sqrt{\frac{2 \log(1/\delta)}{mN}} \right] \\
& \quad + \frac{2\ell_{\text{max}} T}{N} \sum_{i=1}^{N} e_i .
\end{align*}$$

Notice that we need $mN$ to be at least $\tilde{O}(T^2 \log(1/\delta))$, so that $\tilde{\gamma}_N$ and $\ell_{\text{max}} \sqrt{\frac{2 \log(1/\delta)}{mN}}$ are negligible. This leads to the following theorem:

**Theorem A.3.** For our proposed annotation-efficient imitation learning algorithm, with probability at least $1 - \delta$, when $mN$ is $\tilde{O}(T^2 \log(1/\delta))$, there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $J(\hat{\pi}) \leq T [\tilde{\gamma}_N + \frac{2\ell_{\text{max}} T}{N} \sum_{i=1}^{N} e_i] + O(1)$.  

The above shows that the cost of our algorithm can still be bounded in the finite sample setting. Comparing this bound with the bound under the infinite sample setting, we can observe that the bound is still related to $e_i$, the probability that $\hat{\pi}_i$ takes a confident but incorrect action under $d_{\pi^*}$.

### B Implementation Details

#### B.1 Interactive Semantic Parsing Framework

Our system assumes an interactive semantic parsing framework to collect user feedback. In experiments, this is implemented by adapting MISP (Yao et al., 2019b), an open-sourced framework that has demonstrated a strong ability to improve test-time parsing accuracy. In this framework, an agent is comprised of three components: a world model that wraps the base semantic parser and a feedback incorporation module to interpret user feeds and update the semantic parse, an error detector that decides whether to request for user intervention, and an actuator that delivers the agent’s request by asking a natural language question, such that users without domain expertise can understand.

We follow MISP’s instantiation for text-to-SQL tasks to adopt a probability-based uncertainty estimator as the error detector, which triggers user interactions when the probability of the current decision is lower than a threshold. The actuator is instantiated by a grammar-based natural language generator. We use the latest version of MISP that allows multi-choice interactions to improve the system efficiency, i.e., when the parser’s current decision is validated as wrong, the system presents multiple alternative options for user selection. An additional “None of the above options” option is included in case all top options from the system are wrong. Figure 1 shows an example of the user interaction. From there, the system can derive a correct decision to address its uncertainty (e.g., taking “Player” as a WHERE column).

**User Simulator.** Our experiments train each system with simulated user feedback. To this end, we build a user simulator similar to the one used by Yao et al. (2019b), which can access the ground-truth SQL queries. It gives yes/no answer or selects a choice by directly comparing the sampled policy action with the true one in the gold query.

#### B.2 EditSQL Experiment Details

In the data preprocessing step, EditSQL (Zhang et al., 2019) transforms each gold SQL query into a sequence of tokens, where the From clause is removed and each column `Col` is prepended by its paired table name, i.e., `Tab.Col`. However, we

---

6. The actuator is also possible, empirically we found it much slower than the probability-based one and thus is not preferable.

---

8. https://github.com/sunlab-osu/MISP.
observe that sometimes this transformation is not convertible. For example, consider the question “what are the first name and last name of all candidates?” and its gold SQL query: “SELECT T2.first_name, T2.last_name FROM candidates AS T1 JOIN people AS T2 ON T1.candidate_id = T2.person_id”. EditSQL transforms this query into: “select people.first_name, people.last_name”. The transformed sequence accidentally removes the information about table candidates, leading to semantic meaning inconsistent with the question. When using such erroneous sequences as the gold targets in model training, we cannot simulate consistent user feedback, e.g., when the user is asked whether her query is relevant to the table candidates, the simulated user cannot give an affirmative answer given the transformed sequence. To avoid inconsistent user feedback, we remove question-SQL pairs whose transformed sequence is inconsistent with the original SQL query, from the training data. This reduces the size of the training set from 8,421 to 7,377. The validation set is kept untouched for fair evaluation.

The implementation of interactive semantic parsing for EditSQL is the same as Section B.1, except that, in order to cope with the complicated structure of Spider SQL queries, for columns in WHERE, GROUP BY, ORDER BY and HAVING clauses, we additionally provide an option for the user to “remove” the clause, e.g., removing a WHERE clause by picking the “The system does not need to consider any conditions.” option. The confidence threshold $\mu$ is 0.995 as we observe that EditSQL tends to be overconfident.

C Additional Experimental Results

C.1 SQLova Results on Dev set

Figure 4 shows MISP-L’s performance on WikiSQL validation set. We also show in Figure 5 the average number of annotations (i.e., user interactions) per question during the iterative training. Overall, as the base parser is further trained, the system tends to request fewer user interactions. In most cases throughout the training, the system requests no more than one user interaction, demonstrating the annotation efficiency of our algorithm.

C.2 SQLova Results in Theoretical Analysis

As we proved in Section 5, the performance gap between our proposed algorithm and the supervised approach is mainly decided by $\frac{1}{N} \sum_{i=1}^{N} e_i$, an average probability that $\hat{\pi}_i$ makes a confident but wrong decision under $d_{\pi^*}$, (i.e., given a gold partial parse) over $N$ training iterations. More specifically, from our proof of Lemma A.1, $e_i$ can be expressed as:

$$e_i = \frac{1}{T} \sum_{t=1}^{T} \epsilon_{it} = \frac{1}{T} \sum_{t=1}^{T} \epsilon_{it}(1 - \beta_{it}),$$

where $\epsilon_{it}$ denotes policy $\hat{\pi}_i$’s conditional error rate under $d_{\pi^*}$, when it does not query the expert (i.e., being confident about its own action) at step $t$, and $1-\beta_{it}$ denotes the probability that $\hat{\pi}_i$ does not query the expert under $d_{\pi^*}$. $\epsilon_{it}(1 - \beta_{it})$ thus represents a joint probability that $\hat{\pi}_i$ makes confident but wrong action under $d_{\pi^*}$ at step $t$.

To show a reflection of our theoretical analysis on the experiments, we present the values of the following three variables during training: (1) $\tilde{\epsilon}_i = \frac{1}{T} \sum_{t=1}^{T} \epsilon_{it}$, the average value of $\epsilon_{it}$ over $T$ time steps. A smaller $\tilde{\epsilon}_i$ implies a lower conditional error rate and thus a smaller $e_i$ and a smaller performance gap. (2) $\tilde{\beta}_i = \frac{1}{T} \sum_{t=1}^{T} \beta_{it}$, the average value of $\beta_{it}$ over $T$ time steps. A smaller $\tilde{\beta}_i$ (i.e., a larger $1 - \tilde{\beta}_i$) means a smaller probability that $\hat{\pi}_i$ queries the expert (i.e., being more confident). This could lead to a larger $e_i$ and thus a larger performance gap. (3) $e_i$ as defined above. A smaller $e_i$ indicates a smaller performance gap between our algorithm and the supervised approach.

We plot the results in Figure 6. For all initialization settings, we observe that the base parser tends to make more confident actions under a gold partial parse (i.e., decreasing $\tilde{\beta}_i$) when it is trained for more iterations. Meanwhile, the error rate of its confident actions under a gold partial parse is also reduced (i.e., decreasing $\tilde{\epsilon}_i$). When combining the two factors, $e_i$ is shown to keep decreasing, implying that with more iterations that the parser is trained, it gets a tighter cost bound and better performance.

Finally, we notice that a differently initialized parser can end up with a different performance. This is reasonable since a better initialized parser presumably should have a better overall error rate. This is also consistent with our observation in the main experimental results (Section 6).
Figure 4: System parsing accuracy on WikiSQL validation set when they are trained with various numbers of user/expert annotations (top) and for different iterations (bottom). We experiment systems with three initialization settings, using 10%, 5% and 1% of the training data respectively.

Figure 5: Average number of user annotations per question along training iterations (on WikiSQL), when the parser is initialized using 10%, 5% and 1% of training data.

Figure 6: The values of $\tilde{\epsilon}_i$ (a), $\beta_i$ (b) and $e_i$ (c) in MISP-L throughout the training (on WikiSQL), under different initialization settings.