An Imitation Game for Learning Semantic Parsers from User Interaction

Ziyu Yao[1], Yiqi Tang[1], Wen-tau Yih[2], Huan Sun[1], Yu Su[1]
{yao.470, tang.1466, sun.397, su.809}@osu.edu
scottyih@fb.com
[1]The Ohio State University
[2]Facebook AI Research, Seattle

Abstract

Despite the widely successful applications, building a semantic parser is still a tedious process in practice with challenges from costly data annotation and privacy risks. 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 demonstrations 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 demonstrations, we propose a novel *annotation-efficient imitation learning* algorithm, which iteratively collects new datasets by mixing demonstrated states and confident predictions and retrains 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]

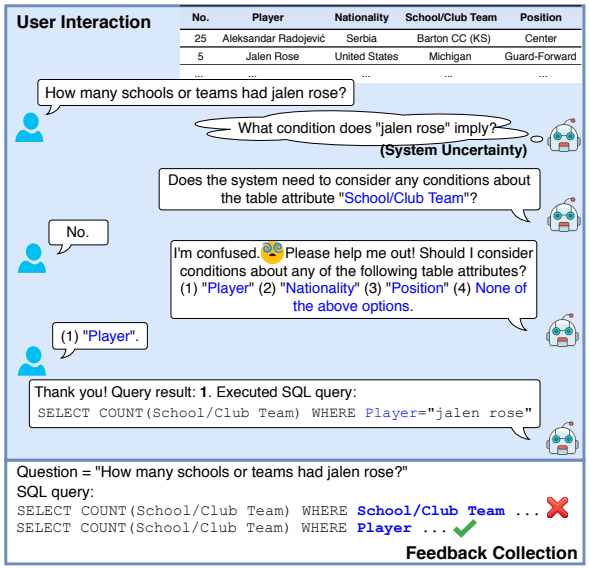


Figure 1: A semantic parser proactively interacts with the user in a friendly way to resolve its uncertainties. In doing so it also gets to imitate the user behavior and continue improving itself autonomously with the hope that eventually it may become as good as the user in interpreting their questions.

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; Yih et al., 2015; Su et al., 2017; Yu et al., 2018). The life cycle 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 the 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 data 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 (Dong et al., 2018) and proac-

[1]Code will be available at https://github.com/sunlab-osu/MISP.

tively prompt for demonstrations from the user, who knows the question best, to resolve them. In doing so, the semantic parser can 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 fully addressed the crucial problem of *how to continually 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; Artzi and Zettlemoyer, 2013; 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 tackle the problem, we propose NEIL, a novel *aNnotation-Efficient Imitation Learning* algorithm for learning semantic parsers from such sparse, fine-grained demonstrations: The agent (semantic parser) only requests for demonstrations when it is uncertain about a state (parsing step). For certain/confident states, actions chosen by the current policy are deemed correct and are executed to continue parsing. The policy is updated iteratively in a Dataset Aggregation fashion (Ross et al., 2011). In 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 state-action pairs for training mitigates the sparsity issue, but it may also introduce training bias. We provide a theoretical analysis and show that, under mild assumptions, the impact of the bias and the quality of the NEIL policy can be controlled by tuning the policy initialization and confidence estimation accuracy.

We also empirically compare NEIL with a number of baselines on the text-to-SQL parsing task. On the WikiSQL (Zhong et al., 2017) dataset, we show that, *when bootstrapped using only 10% of the training data,* NEIL *can achieve almost the same test accuracy (2% absolute loss) as the full expert annotation baseline, while requiring less than 10% of the annotations that the latter needs*, without even taking into account the different unit cost of annotations 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, which provides empirical support for the theoretical analysis. Finally, we demonstrate that NEIL can generalize to more complex semantic parsing tasks such as Spider (Yu et al., 2018).

2 Related Work

Interactive Semantic Parsing. Our work extends the recent idea of leveraging system-user interaction 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; Elgohary et al., 2020; Zeng et al., 2020; Semantic Machines et al., 2020). Gur et al. (2018) built a neural model to identify and correct error spans in generated queries via dialogues. Yao et al. (2019b) formalized a model-based intelligent agent MISP, which enables user interaction via a policy probability-based uncertainty estimator, a grammar-based natural language generator, and a multi-choice question-answer interaction design. More recently, Elgohary et al. (2020) crowdsourced a dataset for fixing incorrect SQL queries using free-form natural language feedback. Semantic Machines et al. (2020) constructed a contextual semantic parsing dataset where agents could trigger conversations to handle exceptions such as ambigu-

ous or incomplete user commands. In this work, we seek to continually improve semantic parsers from such user interaction, a topic that is not carefully studied by the aforementioned work.

Interactive Learning from Feedback. Learning interactively from user feedback has been studied in many NLP tasks (Sokolov et al., 2016; Wang et al., 2016, 2017; Nguyen et al., 2017; Gao et al., 2018; Abujabal et al., 2018; Hancock et al., 2019; Kreutzer and Riezler, 2019). Most relevant to us, Hancock et al. (2019) constructed a self-feeding chatbot that improves itself from user satisfied responses and their feedback on unsatisfied ones.

In the field of semantic parsing, Clarke et al. (2010); Artzi and Zettlemoyer (2013); Iyer et al. (2017) learned semantic parsers from binary user feedback on whether executing the generated query yields correct results. However, often times (especially in information-seeking scenarios) it may not be very practical to expect end users able to validate the denotation correctness (e.g., consider validating an execution result "*103*" for the question "*how many students have a GPA higher than 3.5*" from a massive table). Active learning is also leveraged to 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 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) assumed a "teacher" and actively requested demonstrations for most informative trajectories in the unlabeled data pool. Similar to us, Chernova and Veloso (2009) solicited 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.

Concurrent with our work, Brantley et al. (2020) studied active imitation learning for structured prediction tasks such as named entity recognition. Our work instead focuses on semantic parsing, which presents a unique challenge of *integrality*, i.e., the output sequence (a semantic parse) could only be correct *as a whole* (as opposed to *partially* correct) in order to yield the correct denotation. We therefore propose a new cost function (Section 5) to theoretically analyze the factors that affect the efficacy of learning semantic parsers via imitation.

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 an 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 $s_1 = (q; \ )$ contains only the question. Denote a semantic parser as $\hat{}$, 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* by alternately observing a state and sampling an action from the policy, i.e., $= (s_1, a_1 \sim \hat{}(s_1), ..., s_T, a_T \sim \hat{}(s_T))$, assuming a trajectory length T. The probability of the generated semantic parse becomes: $p_{\hat{}}(a_{1:T}|s_1) = \prod_{t=1}^{T} p_{\hat{}}(a_t|s_t)$.

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, given that it is a principled framework for this purpose and can generalize to various kinds of semantic parsers and logical forms. However, we note that our proposed algorithm is not limited to MISP; it instead depicts a general algorithm for learning semantic parsers from user interaction. We illustrate the application of MISP to a sketch-based parser, SQLova (Hwang et al., 2019), as follows. More details and another example of how it applies to a non-sketch-based parser EditSQL (Zhang et al., 2019) can be found in Appendix B.1.

Example. Consider the SQLova parser, which

generates a query by filling "slots" in a pre-defined SQL sketch "`SELECT Agg SCol WHERE WCol OP VAL`". To complete the SQL query in Figure 1, it first takes three steps: `SCol`="`School/Club Team`" (a_1), `Agg`="`COUNT`" (a_2) and `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 `WCol`="`Player`" (a_3). The parsing continues with `OP`="=" (a_4) and `VAL`="`jalen rose`" (a_5). Here, the trajectory length $T = 5$.

4 Learning Semantic Parsers from User Interaction

In this section, we present NEIL, an *aNnotation-Efficient Imitation Learning* algorithm that trains a parser from user interaction, without requiring a large amount of user feedback (or "annotations"). This property is particularly important for end user-facing systems in practical use. Note that while we apply NEIL to semantic parsing in this work, in principle it can also be applied to other structured prediction tasks (e.g., machine translation).

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). 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*) ,[2] i.e., $a_t \sim$ (s_t). The goal of our algorithm is thus to train policy $\hat{}$ to imitate the expert policy . A general procedure is described in Algorithm 1 (Line 1–9), where $\hat{}$ is learned iteratively for every m user questions. In each iteration, the policy is retrained on an aggregated training data over the past iterations, following the Dataset Aggregation fashion in (Ross et al., 2011).

[2] We follow the imitation learning literature and use "expert" to refer to the imitation target, but the user in our setting by no means needs to be a "domain (SQL) expert".

Algorithm 1 The NEIL Algorithm

Input: Initial training data D_0, policy confidence threshold .
Output: A trained policy $\hat{}$.

1: Initialize $D \leftarrow D_0$.
2: Initialize $\hat{}_1$ 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($; q_j; \hat{}_i;$);
6: Aggregate dataset $D \leftarrow D \cup D_i$;
7: Train policy $\hat{}_{i+1}$ on D using Eq. (1).
8: **end for**
9: **return** best $\hat{}_i$ on validation.
10: **function** PARSE&COLLECT($; q; \hat{}_i;$)
11: Initialize $D_i^{\emptyset} \leftarrow \emptyset$, $s_1 = (q;$).
12: **for** $t = 1$ to T **do**
13: Preview action $a_t = \arg\max_a \hat{}_i(s_t)$;
14: **if** $p_{\hat{}_i}(a_t|s_t) \geq$ **then**
15: $w_t \leftarrow 1$;
16: Collect $D_i^{\emptyset} \leftarrow D_i^{\emptyset} \cup \{(s_t; a_t; w_t)\}$;
17: Execute a_t;
18: **else**
19: Trigger user interaction and derive expert demonstration $a_t \sim$ (s_t);
20: $w_t \leftarrow 1$ if a_t is valid; 0 otherwise;
21: Collect $D_i^{\emptyset} \leftarrow D_i^{\emptyset} \cup \{(s_t; a_t; w_t)\}$;
22: Execute a_t.
23: **end if**
24: **end for**
25: **return** $D_i^{\emptyset}$.
26: **end function**

4.2 Annotation-efficient Imitation Learning

Consider parsing a user question and collecting training data using the parser $\hat{}_i$ 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.

Instead, we propose to adopt an *annotation-efficient* learning strategy in NEIL, which saves user annotations by *selectively* requesting user interactions, as indicated in function PARSE&COLLECT. 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., $p_{\hat{}_i}(a_t|s_t) \geq$

.[3] In this case, the algorithm executes and collects its own action a_t (Line 15–17); otherwise, a system-user interaction will be triggered and the derived demonstration $a_t \sim \quad (s_t)$ will be collected and executed to continue parsing (Line 19–22).

Denote a collected state-action pair as $(s_t; a_t)$, where a_t could be a_t or a_t depending on whether an interaction is requested. To train $\hat{}_{i+1}$ (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:

$$\mathcal{L}(\hat{\pi}_{i+1}) = -\frac{1}{|D|} \sum_{(s_t; a_t; w_t) 2 D} w_t \log p_{\hat{}_i}(\tilde{a}_t | s_t), \qquad (1)$$

where D is the aggregated training data over i iterations and w_t denotes the weight of $(s_t; a_t)$.

We consider assigning weight w_t in three cases: (1) For confident actions a_t, we set $w_t = 1$. This essentially treats the system's own confident actions as gold decisions, which resembles self-training (Scudder, 1965; Nigam and Ghani, 2000; McClosky et al., 2006). (2) For user-confirmed decisions (*valid demonstrations* a_t), such as enforcing a `WHERE` condition on "`Player`" in Figure 1, w_t 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_t, which are identified when the user selects "None of the above options" in Figure 1), we assign $w_t = 0$. 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_t derived from human feedback would be invalid. In this case, the system selects the next available option without further validation and continues parsing.

A possible training strategy to handle case (3) may set w_t 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 the model training. By setting $w_t = 0$, the impact of unaddressed actions is removed from training. 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 NEIL 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 the traditional supervised approach (which trains a policy on full expert annotations, if one could afford that and manage the privacy risk). In this section, we prove that the performance gap between the two approaches is mainly determined by the learning policy's probability of trusting a confident action that turns out to be wrong, which can be controlled in practice.

Our analysis follows prior work (Ross and Bagnell, 2010; Ross et al., 2011) to assume a unified trajectory length T and an infinite number of training samples in each iteration (i.e., $m = \infty$ in Algorithm 1), such that the state space can be full explored by the learning policy. An analysis under the "finite sample" case can be found in Appendix A.5.

5.1 Cost Function for Analysis

Unlike 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.[4] Whenever a policy action is different from the gold one, the whole trajectory will not yield the correct semantic meaning and the parsing is deemed failed. In other words, a well-performing semantic parser should be able to keep staying in the correct trajectory during the parsing. Therefore, for theoretical analysis, we only analyze a policy's performance when it is conditioned on a *gold partial parse*, i.e., $s_t \in d^t$, where d^t is the state distribution in step t when executing the expert policy for first t-1 steps. Let $`(s; \hat{}) = 1 - p_{\hat{}}(a = a\ |s)$ be the loss of $\hat{}$ making a mistake at state s. We define the *cost* (i.e., the inverse *test-time* quality) of a policy as:

$$J(\hat{}) = T \mathsf{E}_{s \ d} \quad `(s; \hat{}) \ ; \qquad (2)$$

where $d \quad = \frac{1}{T} \sum_{t=1}^{T} d^t$ denotes the average expert state distribution (assuming time step t is a

[3] The metric is shown effective for interactive semantic parsing in Yao et al. (2019b). Other confidence measures can also be explored, as we will discuss in Section 7.

[4] 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.

random variable uniformly sampled from $1 \sim T$). A detailed derivation is shown in Appendix A.1.

The *better* a policy $\hat{\pi}$ is, the *smaller* this cost becomes. Note that, by defining Eq. (2), we simplify the analysis from evaluating *the whole trajectory sampled from* $\hat{\pi}$ (as we do in experiments) to evaluating *the expected single-step loss of* $\hat{\pi}$ *conditioned on a gold partial parse*. This cost function makes the analysis easier and meanwhile reflects a consistent relative performance among algorithms for comparison. Next, we compare our NEIL algorithm with the supervised approach by analyzing the upper bounds of their costs.

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 π^*. This gives the policy $\hat{\pi}_{sup}$:

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

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* $\epsilon_N = \min_{\pi \in \Pi} \mathbb{E}_{s \sim d^*}[l(s, \pi)]$, *then* $J(\hat{\pi}_{sup}) = T\epsilon_N$.

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^* as the one being evaluated in the cost function (Eq. (2)).

5.3 Cost Bound of NEIL Algorithm

Recall that, in each training iteration, NEIL samples trajectories by executing actions from both the previously learned policy $\hat{\pi}_i$ and the expert policy π^* (when an interaction is requested). Let π_i denote such a "mixture" policy. We derive the following cost bound of a NEIL policy $\hat{\pi}$:

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

The bound is determined by two terms. The first term $\mathbb{E}_{s \sim d_i}[\ell(s, \hat{\pi}_i)]$ calculates the expected training loss of $\hat{\pi}_i$. Notice that, while the policy is *trained* on states induced by the mixture policy ($s \sim d_i$), what matters to its *test-time* quality is the policy's performance conditioned on a gold partial parse ($s \sim d^*$ in Eq. (2)). This state discrepancy, which does not exist in the supervised approach, explains the performance loss of NEIL, and is bounded by the second term $\ell_{max}\|d_i - d^*\|_1$, the weighted L_1 distance between d_i and d^*. To bound the two terms, we employ a "no-regret" assumption (Kakade and Tewari, 2009; Ross et al., 2011, see Appendix A.3–A.4 for details), which gives the theorem:

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

Here, $\epsilon_N = \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_i}[\ell(s, \pi)]$ denotes the best expected policy loss in *hindsight*, and e_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^*.

We note that a no-regret algorithm requires convexity of the loss function (Hazan et al., 2007; Kakade and Tewari, 2009), which is not satisfied by neural network-based semantic parsers. In general, proving theorems under a non-convex case is not trivial. Therefore, we follow the common practice (e.g., Kingma and Ba (2015); Reddi et al. (2018)) to theoretically analyze the convex case while empirically demonstrating the performance of our NEIL algorithm with non-convex loss functions (i.e., when it applies to neural semantic parsers). More accurate regret bound for non-convex cases will be studied in the future.

Remarks. Compared with the supervised approach (Theorem 5.1), NEIL's cost bound additionally contains a term of $\frac{1}{N}\sum_{i=1}^{N} e_i$, which, as we expect, comes from the aforementioned state discrepancy. Intuitively, if a learning policy frequently executes its own but wrong actions in training, the resulting training states d_i will greatly deviate from the gold ones d^*.

This finding inspires us to restrict the performance gap by reducing the learning policy's error rate *when it does not query the expert*. Empirically this can be achieved by: (1) *Accurate confidence estimation*, so that actions deemed confident are generally correct, and (2) *Moderate policy initialization*, such that in general the policy is less likely to make wrong actions throughout the iterative training. For (1), we set a high confidence threshold δ=0.95, which is demonstrated reliable for MISP (Yao et al., 2019b). We then empirically validate (2) in experiments.

6 Experiments

In this section, we conduct experiments to demonstrate the annotation efficiency of our NEIL algo-

rithm and that it can train semantic parsers for high performance when the parsers are reasonably initialized, which verifies our theoretical analysis.

6.1 Experimental Setup

We compare various systems on the WikiSQL dataset (Zhong et al., 2017). The dataset contains large-scale 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.

We experiment each system with three parser initialization settings, 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 and we simulate user feedback by directly comparing the synthesized query with the gold one. In each iteration, all systems access exactly the same user questions. Depending on how they solicit feedback, each system collects a different number of annotations. At the end of each iteration, we update each system by retraining its parser on its accumulated annotations and the initial training data, and report its *(exact) query match accuracy* on the test set. We also report the *accumulated number of annotations* that each system has requested after each training iteration, in order to compare their annotation efficiency.

In experiments, we consider every 1,000 user questions as one training iteration (i.e., m=1,000 in Algorithm 1). We repeat the whole iterative training for three runs and report average results. *Reproducible details* are included in Appendix B.

6.2 System Comparison

We denote our system as **MISP-NEIL** since it leverages MISP in the back end of NEIL. We compare it with the traditional supervised approach (denoted as **Full Expert**). To investigate the skyline capability of our system, we also present a variant called **MISP-NEIL***, 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-NEIL.

On the other hand, although execution feedback-based learning systems (Clarke et al., 2010; Artzi and Zettlemoyer, 2013; Iyer et al., 2017) may not be very practical for end users, we include them nonetheless in the interest of a comprehensive comparison. This leads to two baselines. 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 results to further improve the parser. The **Binary User+Expert** system additionally collects full expert SQL annotations when the generated SQL queries do not yield correct answers.

Given the completely different nature of annotations from Binary User (which validate the *denotation*) and those from Full Expert and MISP-NEIL (which validate a semantic parse's *constituents*), there may not exist a universally fair way to convert one's annotation consumption into the other's. Therefore, in the following sections, we only present and discuss Binary User(+Expert) in terms of their parsing accuracy under different training iterations. To give an estimation of their annotation efficiency for reference, we design a compromised annotation calculation metric for Binary User(+Expert) and include their results on WikiSQL validation set in Appendix C.

Finally, while our MISP-NEIL and the aforementioned baselines all leverage feedback from users or domain experts, an interesting question is how much gain they could obtain compared with using no annotation or feedback at all. To this end, we compare the systems with a **Self Train** baseline (Scudder, 1965; Nigam and Ghani, 2000; McClosky et al., 2006). In each iteration, this baseline collects SQL queries generated *by itself* as the new gold annotations for further training. We additionally apply a confidence threshold to improve the collection quality, i.e., only SQL queries with probability $p_\wedge(a_{1:T}|s_1)$ greater than 0.5 are included. This strategy empirically leads to better performance. Intuitively, we expect Self Train to perform no better than any other systems in our experiments, since no human feedback is provided to correct mistakes in its collection.

6.3 Experimental Results

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

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

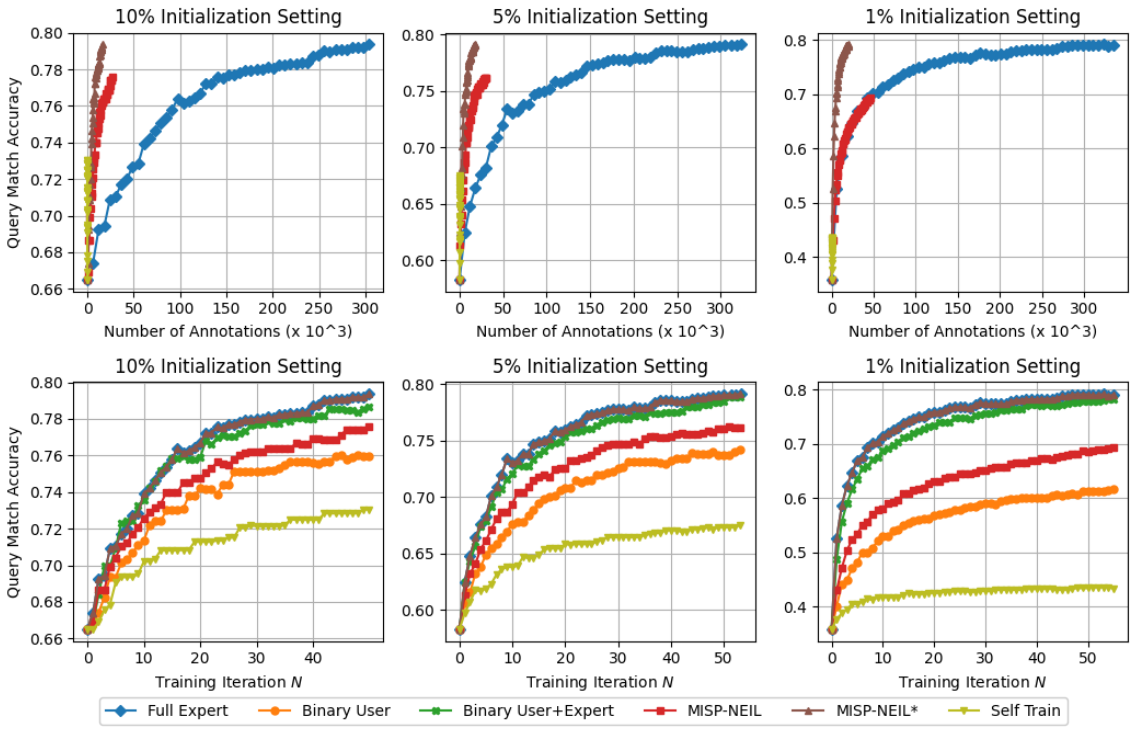


Figure 2: Parsing accuracy on WikiSQL test set when systems are trained with various numbers of user/expert annotations (top) and for different iterations (bottom). We experiment with three initialization settings, using 10%, 5% and 1% of the training data respectively. Results on validation set can be found in Appendix C.

- *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 that a system requires to train a parser. For Full Expert, this number equals the trajectory length of the gold query (e.g., 5 for the query in Figure 1); for MISP-NEIL and MISP-NEIL*, it is the number of user interactions during training. Note that while we do not differentiate the actual (e.g., time/financial) cost of users from that of 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. For Self Train, the number of annotations is always zero since it does not request any human feedback for the online user questions.

Our results in Figure 2 (top) demonstrate that MISP-NEIL consistently consumes a comparable or smaller amount of annotations to train the parser to reach the same parsing accuracy. Figure 5 in Appendix further shows that, on average, it requires no more than one interaction for each user question along the training. Particularly in the 10% initialization setting, MISP-NEIL uses less than 10% of the total annotations that Full Expert needs in the end. Given the limited size of WikiSQL training set, the simulation experiments currently can only show MISP-NEIL'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 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-NEIL 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.2, we plot the value of e_i, the probability that $\wedge_i$ 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.

Our system also surpasses Binary User. We find

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-NEIL circumvents this issue by directly validating the *semantic meaning* of intermediate parsing decisions. The performance of Binary User+Expert is close to Full Expert as it has additionally involved expert annotations on a considerable number of user questions, which on the other hand also leads to extra annotation overhead.

When it is assumed with perfect interaction design and confidence estimator, MISP-NEIL* shows striking superiority in both aspects. Since it always corrects wrong decisions immediately, MISP-NEIL* can collect and derive the same training examples as Full Expert, and thus trains the parser to Full Expert's performance level in Figure 2 (bottom). However, it requires only 6% of the annotations that Full Expert needs (Figure 2, top). These observations imply large room for MISP-NEIL to be improved in the future.

Finally, we observe that all feedback-based learning systems outperform Self Train dramatically (Figure 2, bottom). This verifies the benefit of learning from human feedback.

6.4 Generalize to Complex SQL Queries

We next investigate whether MISP-NEIL can generalize to the complex SQL queries in the Spider dataset (Yu et al., 2018), which 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 pairs for training after data cleaning; see Appendix B.3 for details), we only experiment with one initialization setting, using 10% of the training set. Since EditSQL does 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 Full Expert and Self Train. Parsers are evaluated on Spider Dev set since its test set is not publicly available.

Figure 3 (top) shows that MISP-NEIL and MISP-NEIL* consistently achieve comparable or better annotation efficiency while enjoying the advantage of learning from end user interaction. We expect this superiority to continue as the systems receive more user questions beyond Spider. Meanwhile,

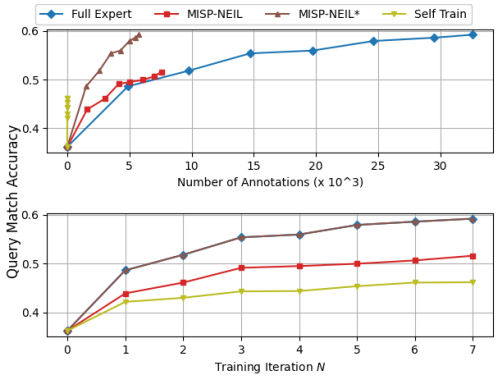


Figure 3: Parsing accuracy on Spider Dev set when systems are trained with various numbers of user/expert annotations and for different iterations.

we also notice that the gain is smaller and MISP-NEIL suffers from a large performance loss compared with Full Expert (Figure 3, bottom), due to the poor parser initialization and the SQL query complexity. This can be addressed via adopting better interaction designs and more accurate confidence estimation, as shown by MISP-NEIL*. Similarly as in WikiSQL experiments, Self Train performs worse than human-in-the-loop learning systems, as there is no means to correct wrong predictions in its collected annotations.

7 Conclusion and Future Work

Our work shows the possibility of continually learning semantic parsers from fine-grained end user interaction. As a pilot study, we experiment systems with simulated user interaction. One important future work is thus to conduct large-scale user studies and train parsers from real user interaction. This is not trivial and has to account for uncertainties such as noisy user feedback. We also plan to derive a more realistic formulation of user/expert annotation costs by analyzing real user statistics (e.g., average time spent on each question).

In experiments, we observe that neural semantic parsers tend to be overconfident and training them with more data does not mitigate this issue. In the future, we will look into more accurate confidence measure via neural network calibration (Guo et al., 2017) or using machine learning components (e.g., answer triggering (Zhao et al., 2017) or a reinforced active selector (Fang et al., 2017)).

Finally, we believe our algorithm can be applied to save annotation effort for other NLP tasks, especially the low-resource ones (Mayhew et al., 2019).

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. The research conducted by the Ohio State University authors was sponsored in part by the Army Research Office under cooperative agreements W911NF-17-1-0412, NSF Grant IIS1815674, NSF CAREER #1942980, 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. The Spider dataset (Yu et al., 2018) is distributed under the CC BY-SA 4.0 license.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-ending learning for open-domain question answering over knowledge bases. In *Proceedings of the 2018 World Wide Web Conference*, pages 1053–1062.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

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.

Kianté Brantley, Amr Sharaf, and Hal Daumé, III. 2020. Active imitation learing with noisy guidance. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Ohio Supercomputer Center. 1987. Ohio supercomputer center. 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 Language Processing (Volume 2: Short Papers)*, pages 175–180.

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.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip R Cohen, and Mark Johnson. 2018. Active learning for deep semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 43–48.

Ahmed Elgohary, Saghar Hosseini, and Ahmed Hassan Awadallah. 2020. Speak to your parser: Interactive text-to-SQL with natural language feedback. In *Annual Conference of the Association for Computational Linguistics (ACL 2020)*.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605.

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.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3667–3684.

Elad Hazan, Amit Agarwal, and Satyen Kale. 2007. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2337–2342.

Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

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

Kshitij Judah, Alan P Fern, Thomas G Dietterich, and Prasad Tadepalli. 2014. Active imitation learning: Formal and practical reductions to iid learning. *Journal of Machine Learning Research*, 15:4105–4143.

Sham M Kakade and Ambuj Tewari. 2009. On the generalization ability of online strongly convex programming algorithms. In *Advances in Neural Information Processing Systems*, pages 801–808.

Beomjoon Kim and Joelle Pineau. 2013. Maximum mean discrepancy imitation learning. *Robotics: Science and systems*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Julia Kreutzer and Stefan Riezler. 2019. Self-regulated interactive sequence-to-sequence learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 303–315.

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

Natasha Lomas. 2019. Google ordered to halt human review of voice AI recordings over privacy risks. https://techcrunch.com/2019/08/02/google-ordered-to-halt-human-review-of-voice-ai-recordings-over-privacy-risks/. Accessed: 2020-04-28.

Stephen Mayhew, Snigdha Chaturvedi, Chen-Tse Tsai, and Dan Roth. 2019. Named entity recognition with partially annotated training data. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 645–655.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1464–1474.

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 Matusov. 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.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations*.

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.

H Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin

Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

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, Miaosen 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.

Sida I Wang, Samuel Ginn, Percy Liang, and Christopher D Manning. 2017. Naturalizing a programming language via interactive learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 929–938.

Sida I Wang, Percy Liang, and Christopher D Manning. 2016. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2368–2378.

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.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331.

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.

Jichuan Zeng, Xi Victoria Lin, Steven CH Hoi, Richard Socher, Caiming Xiong, Michael Lyu, and Irwin King. 2020. Photon: A robust cross-domain text-to-SQL system. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 204–214.

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.

Jie Zhao, Yu Su, Ziyu Guan, and Huan Sun. 2017. An end-to-end deep framework for answer triggering with a novel group-level objective. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1276–1282.

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 bounds of the supervised approach and our proposed NEIL algorithm (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), such that the state space in training can be full explored by the learning policy.

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 NEIL algorithm. Finally, in Appendix A.5, we show the cost bound of NEIL 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 and the parsing is deemed failed. Therefore, we analyze a policy's performance only when it is conditioned on *a gold partial parse*. Intuitively, a policy with better quality should have a higher parsing accuracy under a gold partial parse, so that it is more likely to sample a completely correct trajectory in inference time.

Given a question q and denoting $a_{1:t}$ as the gold partial trajectory sampled by the expert policy , we first define the *cost* of sampling a partial trajectory $a_{1:t} = (a_1; \ldots; a_t)$ as:

$$C(q; a_{1:t}) = \begin{cases} 0 & \text{if } a_{1:t} = a_{1:t} \\ 1 & \text{otherwise} \end{cases}.$$

In other words, a sampled partial trajectory is correct if and only if it is the same as the gold partial parse. Based on this definition, we further define the expected cost of policy $\hat{}$ *in a single time step* t (given the question q) as:

$$\begin{aligned} C^t_{\hat{}}(q) &= \mathrm{E}_{a_{1:t\ 1}\ \ \mathrm{E}_{a_t\ \hat{}}}[C(q; a_{1:t})] \\ &= \mathrm{E}_{a_{1:t\ 1}\ \ [1 - p_{\hat{}}(a_t = a_t | q; a_{1:t\ 1})]}. \end{aligned}$$

Here, $a_{1:t\ 1} \sim$ denotes a gold partial parse till the (t-1)-th step, which is obtained by executing the expert policy for the first t-1 steps (given q), and $p_{\hat{}}(a_t = a_t | q; a_{1:t\ 1})$ denotes the probability that $\hat{}$ samples action a_t given a question q and a partial parse $a_{1:t\ 1}$. By taking an expectation over all questions $q \in \mathcal{Q}$, we have the following derivations:

$$\begin{aligned} \mathrm{E}_{q \sim Q}[C^t_{\hat{}}(q)] &= \mathrm{E}_{q \sim Q; a_{1:t\ 1}\ \ [1\ \ p_{\hat{}}(a_t = a_t | q; a_{1:t\ 1})]} \\ &= \mathrm{E}_{s_t\ \ d^t\ \ [1\ \ p_{\hat{}}(a_t = a_t | s_t)]}. \end{aligned}$$

The second equality holds by the definition $s_t = (q; a_{1:t\ 1})$, and d^t is the "expert state distribution" in step t when executing the expert policy for first t-1 steps. In this analysis, we follow Ross and Bagnell (2010); Ross et al. (2011) to assume a unified decision length T. By summing up the above expected cost over the T steps, we define the *cost* (i.e., the inverse *test-time* quality) of policy $\hat{}$:

$$\begin{aligned} J(\hat{}) &= \sum_{t=1}^{T} \mathrm{E}_{q \sim Q}[C^t_{\hat{}}(q)] \\ &= \sum_{t=1}^{T} \mathrm{E}_{s_t\ \ d^t\ \ [1 - p_{\hat{}}(a_t = a_t | s_t)]}. \end{aligned}$$

Denote $\grave{}(s; \hat{}) = 1 - p_{\hat{}}(a = a | s); a \sim \hat{}(s); a \sim (s)$ as the "single-step loss function", which is bounded within $[0; 1]$, then the cost of policy $\hat{}$ can be simplified as:

$$\begin{aligned} J(\hat{}) &= \sum_{t=1}^{T} \mathrm{E}_{s_t\ \ d^t\ \ \grave{}(s_t; \hat{})} \\ &= T\, \mathrm{E}_{t\ \ U(1;T)} \mathrm{E}_{s_t\ \ d^t\ \ \grave{}(s_t; \hat{})} \\ &= T\, \mathrm{E}_{s\ \ d\ \ \grave{}(s; \hat{})}; \end{aligned} \tag{3}$$

where $d = \frac{1}{T} \sum_{t=1}^{T} d^t$ 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).

A.2 Cost Bound of Supervised Approach

In this section, we analyze the cost bound of the supervised approach. Recall that the supervised approach trains a policy $\hat{}$ using the standard supervised learning algorithm with supervision from at every decision step. Therefore, it finds the best policy $\hat{}_{sup}$ on infinite samples as:

$$\hat{}_{sup} = \arg\min_{2} \mathrm{E}_{s\ \ d\ \ [\grave{}(s;\ \ )]}; \tag{4}$$

where denotes the policy space induced by the model architecture, and the expectation over s is

sampled from the whole d state space because of the "infinite sample" assumption. The supervised approach thus obtains the following cost bound:

$$J(\hat{\pi}_{sup}) = T\mathbb{E}_{s\sim d}[\ell(s,\hat{\pi}_{sup})] = T\min_{\pi\in\Pi}\mathbb{E}_{s\sim d}[\ell(s,\pi)].$$

This gives the following theorem:

Theorem A.1. *For supervised approach, let* $\epsilon_N = \min_{\pi\in\Pi}\mathbb{E}_{s\sim d}[\ell(s,\pi)]$, *then* $J(\hat{\pi}_{sup}) = T\epsilon_N$.

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 (given the "infinite sample" assumption) as in evaluation (Eq. (3)). As we will show next, the proposed NEIL algorithm breaks this consistency while enjoying the benefit of high annotation efficiency, which explains the performance gap.

A.3 No-regret Assumption

Before showing the cost bound of our NEIL algorithm, we introduce a "no-regret" assumption (Kakade and Tewari, 2009; Ross et al., 2011) that is leveraged in the derivation.

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

$$\frac{1}{N}\sum_{i=1}^{N}\ell_i(\hat{\pi}_i) - \epsilon_N \le \gamma_N$$

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

This assumption characterizes an important policy learning pattern: As a policy is trained for an infinite number of iterations, on average, its expected training loss ($\frac{1}{N}\sum_{i=1}^{N}\ell_i(\hat{\pi}_i)$) will converge to the loss of the best policy in hindsight (ϵ_N). In our scenario, this assumption implies that, while our policy is trained on online labels from both the expert policy (when it is queried) and the previously learned policy $\hat{\pi}_i$ (when the agent is confident), it still gradually fits to the best policy over the same state space in training (d_{β_i}). In other words, the likely noisy labels from $\hat{\pi}_i$ do not harm the model fitting to the expert policy in general.

Many no-regret algorithms (Hazan et al., 2007; Kakade and Tewari, 2009) that guarantee $\gamma_N \in O(\frac{1}{N})$ require convexity or strong-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 general, proving theorems under a non-convex case is not trivial. In this analysis, we follow the common practice (see Kingma and Ba (2015); Reddi et al. (2018) for example) to theoretically analyze the convex case while empirically demonstrating the non-convex case. A more accurate regret bound for non-convex neural networks (which may result in a slower γ_N convergence speed with respect to N) can be studied in the future.

A.4 Cost Bound of NEIL Algorithm

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

$$\begin{aligned} J(\hat{\pi}) &= \min_{\hat{\pi}\in\hat{\pi}_{1:N}} T\mathbb{E}_{s\sim d_{\hat{\pi}}}[\ell(s,\hat{\pi})] \\ &\le \frac{T}{N}\sum_{i=1}^{N}\mathbb{E}_{s\sim d_{\hat{\pi}_i}}[\ell(s,\hat{\pi}_i)] \\ &\le \frac{T}{N}\sum_{i=1}^{N}\mathbb{E}_{s\sim d_{\beta_i}}[\ell(s,\hat{\pi}_i)] + \ell_{max}\|d_{\beta_i} - d_{\hat{\pi}_i}\|_1 . \end{aligned} \tag{5}$$

From the last inequality, we can see that the cost bound of NEIL is restricted by two terms. The first term $\mathbb{E}_{s\sim d_{\beta_i}}[\ell(s,\hat{\pi}_i)]$ denotes the expected loss of $\hat{\pi}_i$ under the states induced by β_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 $\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{s\sim d_{\beta_i}}[\ell(s,\hat{\pi}_i)] \le \epsilon_N + \gamma_N$. Here, $\epsilon_N = \min_{\pi\in\Pi}\frac{1}{N}\sum_{i=1}^{N}\ell_i(\pi)$ denotes the best expected training loss in hindsight.

The second term denotes the L_1 distance between state distributions induced by β_i and $\hat{\pi}_i$, weighted by the maximum loss value l_{max} that $\hat{\pi}_i$ encounters over the training. As we notice, unlike the supervised approach, NEIL *trains* a policy under d_{β_i}, while what matters to its *test-time* quality is its performance on the state distribution $d_{\hat{\pi}}$ (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 $\hat{\pi}_i$ makes a confident (i.e., without querying the expert policy) but wrong action under d, and introduce the following lemma:

Lemma A.1. $\|d_{\hat{\pi}_i} - d\|_1 \leq 2Te_i$.

Proof. Let ϵ_{it} be the probability of querying the expert policy under d^t, $\epsilon_{\hat{\pi}_i t}$ the error rate of $\hat{\pi}_i$ under d^t (w.r.t. π), and d' any state distribution besides d. We can then express $d_{\hat{\pi}_i}$ by:

$$d_{\hat{\pi}_i} = \prod_{t=1}^{T}(\epsilon_{it} + (1-\epsilon_{it})(1-\epsilon_{\hat{\pi}_i t}))d + (1 - \prod_{t=1}^{T}(\epsilon_{it} + (1-\epsilon_{it})(1-\epsilon_{\hat{\pi}_i t})))d'.$$

The distance between $d_{\hat{\pi}_i}$ and d thus becomes

$$\begin{aligned}
&\|d_{\hat{\pi}_i} - d\|_1 \\
=&(1 - \prod_{t=1}^{T}(\epsilon_{it} + (1-\epsilon_{it})(1-\epsilon_{\hat{\pi}_i t})))\|d - d'\|_1 \\
\leq&2(1 - \prod_{t=1}^{T}(\epsilon_{it} + (1-\epsilon_{it})(1-\epsilon_{\hat{\pi}_i t}))) \\
\leq&2\sum_{t=1}^{T}[1 - (\epsilon_{it} + (1-\epsilon_{it})(1-\epsilon_{\hat{\pi}_i t}))] \\
\leq&2\sum_{t=1}^{T}[\epsilon_{\hat{\pi}_i t}(1-\epsilon_{it})] \\
\leq&2\sum_{t=1}^{T} e_{it} \\
=&2Te_i.
\end{aligned}$$

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:

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

Given a large enough N ($N \in \mathcal{O}(T)$), 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 the proposed* NEIL *algorithm, if N is $\mathcal{O}(T)$, there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ s.t. $J(\hat{\pi}) \leq T\epsilon_N + \frac{2T\ell_{max}}{N}\sum_{i=1}^{N} e_i + O(1)$.*

By comparing Theorem A.1 and Theorem A.2, it is obvious that the performance gap between NEIL and the supervised approach is bounded by the term around $\frac{1}{N}\sum_{i=1}^{N} e_i$. We discuss its implications in Section 5 and show that, in practice, this performance gap can be controlled by carefully initializing the policy and choosing a more accurate confidence estimator.

Discussion about MISP-NEIL*. In experiments, we consider a skyline instantiation of NEIL, called MISP-NEIL*. This instantiation is assumed with perfect confidence estimation and interaction design, such that it can precisely detect and correct its intermediate mistakes during parsing. Therefore, MISP-NEIL* presents an upper bound performance (i.e., the *tightest* cost bound) of NEIL. This can be interpreted theoretically. In fact, for MISP-NEIL*, 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, $d_{\hat{\pi}_i} = d$, so $\epsilon_N = \min_{\hat{\pi}} \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{s\sim d}[l(s, \hat{\pi})] = \min_{\hat{\pi}} \mathbb{E}_{s\sim d}[l(s, \hat{\pi})]$. Therefore, according to Theorem A.2, MISP-NEIL* has a cost bound of:

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

where $\epsilon_N = \min_{\hat{\pi}} \mathbb{E}_{s\sim d}[l(s, \hat{\pi})]$.

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

A.5 Cost Bound of NEIL Algorithm in Finite Sample Case

The theorem in the previous section holds when the algorithm observes infinite trajectories in training. However, in practice, NEIL observes the training loss from only a finite set of m trajectories in each iteration. For this consideration, in the following discussion, we provide a proof of the cost bound of NEIL under the finite sample case.

Denote D_i as the m trajectories collected in the i-th iteration and $\ell_i(\hat{\pi}_i) = \mathbb{E}_{s\sim D_i}[\ell(s, \hat{\pi}_i)]$. Applying the no-regret assumption (Assumption A.1) allows us to bound the average expected policy training loss: $\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{s\sim D_i} \ell(s, \hat{\pi}_i) - \epsilon_N \leq \epsilon'_N$,

where $\epsilon_N = \min_{\hat{\pi}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim D_i}[\ell(s, \hat{\pi})]$ denotes the loss of the best policy in hindsight on the finite samples.

Following Eq. (5), we need to switch the derivation from the expected loss of $\hat{\pi}_i$ over d_{β_i} (i.e., $\mathbb{E}_{s \sim d_{\beta_i}}[\ell(s, \hat{\pi}_i)]$) to that over D_i (i.e., $\mathbb{E}_{s \sim D_i}[\ell(s, \hat{\pi}_i)]$), 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_{β_i} and the average loss of $\hat{\pi}_i$ under the j-th sample trajectory with β at iteration i. The random variables Y_{ij} over all $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, m\}$ are all zero mean, bounded in $[-\ell_{max}, \ell_{max}]$ and form a martingale in the order of $Y_{11}, Y_{12}, \ldots, Y_{1m}, Y_{21}, \ldots, Y_{Nm}$. By Azuma-Hoeffding's inequality (Azuma, 1967; Hoeffding, 1994), $\frac{1}{mN} \sum_{i=1}^{N} \sum_{j=1}^{m} Y_{ij} \leq \ell_{max} \sqrt{\frac{2 \log(1/\delta)}{mN}}$ 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:

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

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

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

The theorem shows that the cost of NEIL 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 $d_{\hat{\pi}}$.

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[5] 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).

As a general interactive semantic parsing framework, MISP has its advantage of being generalizable to different kinds of semantic parsers (as long as their parsing process can be formulated as taking a sequence of actions in their respective action space) and various logical forms (e.g., lambda expressions). Although it could be non-trivial to instantiate such an interactive system, we note that it is a one-time effort for all datasets of the same logical form.

Example of Non-sketch-based Parsers. In addi-

[5] https://github.com/sunlab-osu/MISP.

tion to the example of the SQLova parser (Hwang et al., 2019) that we provide in Section 3, here we show how the EditSQL parser (Zhang et al., 2019) is formulated under MISP. Unlike SQLova, EditSQL does not assume any SQL sketch; it instead generates a SQL query "token by token".[6] Consider the SQL query in Figure 1. EditSQL takes actions: a_1="`SELECT`", a_2="`COUNT`", a_3="`(`", a_4="`School/Club Team`", a_5="`)`", etc. Therefore, the action space of EditSQL consists of all SQL keywords, grammatical constituents (e.g., "`(`"), and available table columns. In this case, MISP only validates semantically meaningful actions (including aggregators, operators, column names, etc.) while skipping others (including trivial symbols like "`(`" and most SQL keywords[7]).

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) in MISP, 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. When the true option is not presented within the system provided choices, the user is simulated to select "None of the above options".

B.2 WikiSQL Experiment Details

Dataset&Model. Our main experiments consider the WikiSQL benchmark dataset (Zhong et al., 2017),[8] which contains 56,355/8,421/15,878 question-SQL query pairs in the training/validation/test set. We use exactly the same data split as Zhong et al. (2017).

We choose SQLova (Hwang et al., 2019), one of the open-sourced[9] top-performing semantic parser on WikiSQL, as the base parser, which ensures reasonable model capability to study continual learning. Hyper-parameters are set the same as the ones recommended by the SQLova authors on their GitHub repository,[10] except that we use a learning rate of 1e-5 for fine-tuning the BERT model. Empirically we found out this relatively larger learning rate can greatly accelerate the model learning without affecting the model performance significantly. The total number of model parameters is around 118M, with 110M from BERT-Base (Uncased)[11] and 8M from the SQLova parser side.

Early stop is used to accelerate model training in each training iteration. Specifically, we stop model training if it does not show improvement on the validation set for a consecutive number of epochs. We set this number to 10 before the 30-th training iteration when the total training data is in a relatively small size, and decay it to 5 after the 30-th iteration. We follow SQLova when preprocessing the WikiSQL data.

Experimental Setup. We study a "continual learning" problem and experiment various systems with three initialization settings, as suggested by our theoretical analysis (Section 5). Specifically, we use 10% (5,636 pairs), 5% (2,818 pairs), and 1% (564 pairs) of the total training data for parser initialization, respectively.

In each initialization setting, the remaining training data is used to simulate user questions that a system receives after deployment. The user questions come in a random order. We repeat three random runs (i.e., three random orders of user questions) and report the average system performance. Notice that, we ensure each system receive the same user question (but may have different user feedback depending on their interaction designs) during iterative training, for a fair comparison. Systems update (retrain) their base semantic parsers periodically for every 1,000 user questions.

Metrics. In the end of each iteration, we evaluate the system's performance, including:

- Parsing accuracy. We measure the query match accuracy (i.e., logical form accuracy) using the script from SQLova implementation.
- An accumulated number of user/expert annotations (introduced in Section 6.3). Different systems request different kinds of user/expert annotations. Therefore, even when serving the user on the same user question, different systems require different numbers of annotations. This metric sums up the total number of annotations that each system has requested after each training iteration.

Calculating the aforementioned metrics allow us to plot Figure 2 and Figure 4.

[6] EditSQL considers a column name as a single "token", although it may actually contain several words (e.g., `School/Club Team`).

[7] Except `WHERE`, `GROUP BY`, `ORDER BY` and `HAVING`; see Appendix B.3 for details.

[8] https://github.com/salesforce/WikiSQL.

[9] https://github.com/naver/sqlova.

[10] https://github.com/naver/sqlova#running-code.

[11] https://github.com/google-research/bert#pre-trained-models.

Compute. We complete experiments on Nvidia GeForce RTX 2080Ti (11GB). Models are all implemented using PyTorch.[12] The run time for each training iteration varies depending on the accumulated training data size. To finish the 50+ iterations of (re-)training, each system takes around 15 days. In the weak 1% initialization case, the Binary User baseline takes less time (around 10 days), since most of its predicted queries are wrong and thus are not included into its training data.

B.3 EditSQL Experiment Details

Data&Model. The Spider dataset (Yu et al., 2018) contains 8,421 question-SQL pairs for training and 1,034 pairs for validation.[13] The test set is not publicly available and is thus not used in our experiments.

We choose EditSQL (Zhang et al., 2019) as the base semantic parser,[14] since it is one of the open-sourced state-of-the-art models on Spider. All hyper-parameters are set following (Zhang et al., 2019). Pre-trained BERT model is also used. Totally there are around 120M parameters in the model, with 110M from the BERT-Base (Uncased)[15] and 10M from the EditSQL parser side. Early stop is additionally used to accelerate model training. Specifically, we stop model training when it does not show improvement on validation for 5 consecutive epochs.

In the data preprocessing step, EditSQL 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 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` in the original SQL query, 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 based on the transformed sequence. To avoid inconsistent user feedback, we remove question-SQL pairs whose transformed sequence is inconsistent with the original gold SQL query, from the training data. This can be easily done by using EditSQL's post-processing script to convert a preprocessed sequence back to the SQL format. Only when the converted query is the same as the original one, the transformation is consistent. 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. We also adjust the "semantic unit" definition in MISP[16] to deal with the autoregressive decoding of EditSQL. For example, instead of asking first about a `SELECT` column and then about its aggregator, we define one semantic unit to inquire about both the column and its aggregator.

To instantiate NEIL, the confidence threshold is 0.995 as we observe that EditSQL tends to be overconfident.

Experimental Setup. We experiment with one initialization setting, using 10% of the total training data (i.e., 737 question-SQL pairs), and systems update (retrain) their base semantic parsers periodically for every 1,000 user questions as in WikiSQL experiments. We report system performance averaged over three random runs (i.e., three random orders of user questions).

We also tried using more training data for initialization. However, since the total training data in Spider is very limited in size, more initialization data means fewer data for simulating online

[12] https://pytorch.org/.

[13] https://github.com/taoyds/spider.

[14] https://github.com/ryanzhumich/editsql.

[15] https://github.com/google-research/bert#pre-trained-models.

[16] https://github.com/sunlab-osu/MISP/blob/multichoice_q/MISP_SQL/tag_seq_logic.md.

user questions and conducting continual learning. This leads to less clear experimental observations (e.g., even the Full Expert system shows fluctuation, probably due to data redundancy or an issue with model architecture capability). Therefore, we only focus on the 10% initialization setting.

Metrics. We measure each system similarly as in WikiSQL experiments. For parsing performance, we calculate the exact match accuracy using scripts from the EditSQL implementation.

Compute. We complete experiments on Nvidia GeForce RTX 2080Ti (11GB). Models are implemented using PyTorch. The run time for each training iteration varies depending on the accumulated training data size. Finishing the whole iterative learning takes around 5 days for all systems.

C Additional Experimental Results

C.1 Additional SQLova Results

Figure 4 shows different systems' performance on WikiSQL validation set. For Binary User(+Expert), it is hard to quantify "one annotation", which varies according to the actual database size and the query difficulty. As a compromise, we approximate this number by calculating it in the same way as Full Expert, with the assumption that in general validating execution results is as hard as validating the SQL query itself.

We also show in Figure 5 the average number of annotations (i.e., user interactions) that MISP-NEIL requires per question during the iterative training. Overall, as the base parser is further trained, our 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 NEIL algorithm.

C.2 Connection to Theoretical Analysis

As we proved in Section 5, the performance gap between our proposed NEIL algorithm and the supervised approach is mainly decided by $\frac{1}{N}\sum_{i=1}^{N} e_i$, an average probability that $\hat{}_i$ makes a confident but wrong decision under d (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:

$$e_i = \frac{1}{T}\sum_{t=1}^{T} e_{it} = \frac{1}{T}\sum_{t=1}^{T} \epsilon_{it}(1-\delta_{it});$$

where ϵ_{it} denotes policy $\hat{}_i$'s conditional error rate under d^t when it does not query the expert (i.e., being confident about its own action) at step t, and $1-\delta_{it}$ denotes the probability that $\hat{}_i$ does not query the expert under d^t. $\epsilon_{it}(1-\delta_{it})$ thus represents a joint probability that $\hat{}_i$ makes confident but wrong action under d^t 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) $\epsilon_i = \frac{1}{T}\sum_{t=1}^{T}\epsilon_{it}$, the average value of ϵ_{it} over T time steps. A smaller ϵ_i implies a lower conditional error rate and thus a smaller e_i and a smaller performance gap. (2) $\delta_i = \frac{1}{T}\sum_{t=1}^{T}\delta_{it}$, the average value of δ_{it} over T time steps. A smaller δ_i (i.e., a larger $1-\delta_i$) means a smaller probability that $\hat{}_i$ 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 previously. A smaller e_i indicates a smaller performance gap between our algorithm and the supervised approach.

We plot the results of our MISP-NEIL system (based on SQLova) 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 δ_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 ϵ_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 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 6.3).

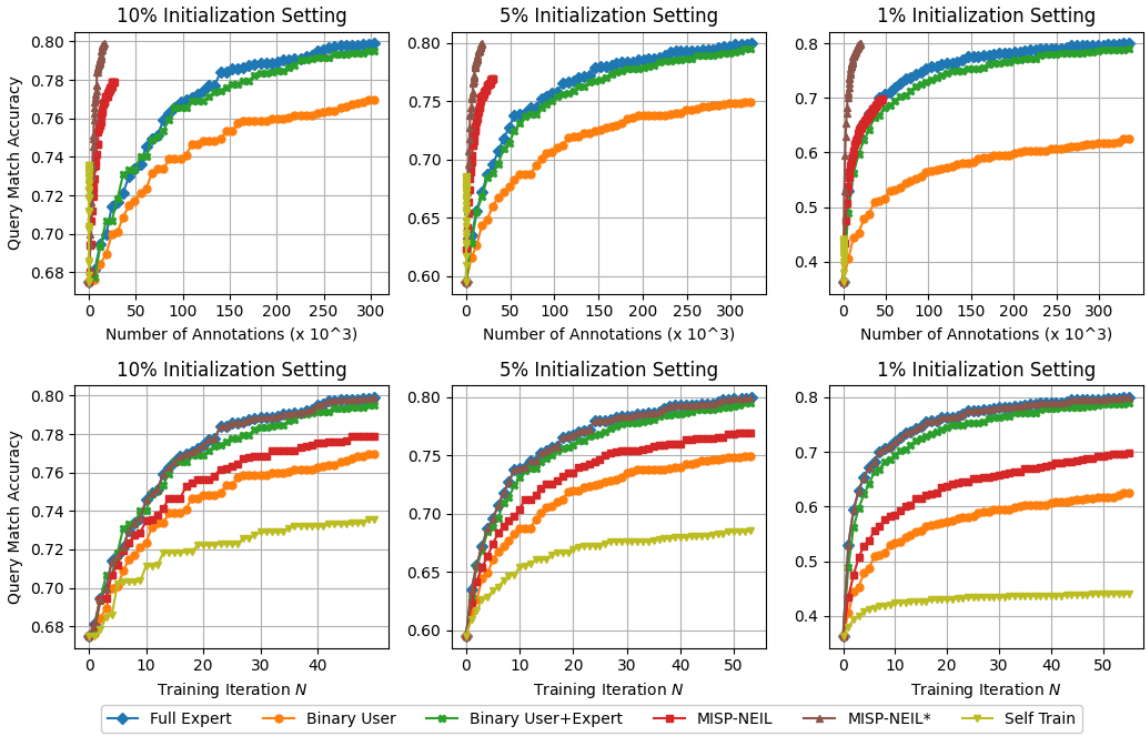


Figure 4: Parsing accuracy on WikiSQL validation set when systems are trained with various numbers of user/expert annotations (top) and for different iterations (bottom). We experiment systems with three initialization settings, using 10%, 5% and 1% of the training data respectively.

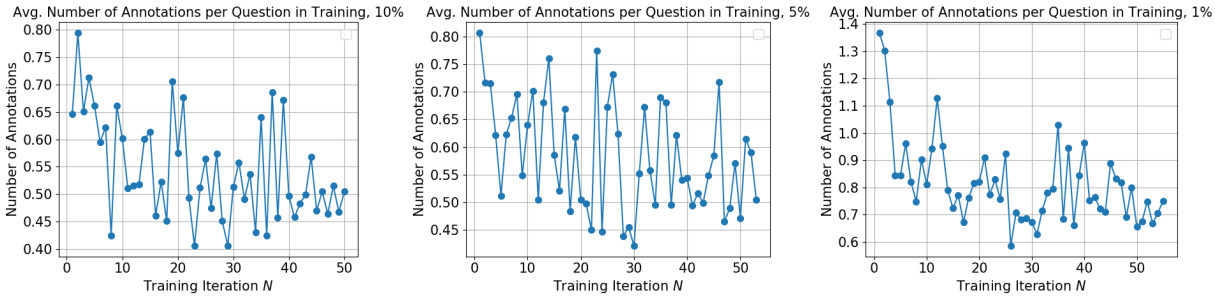
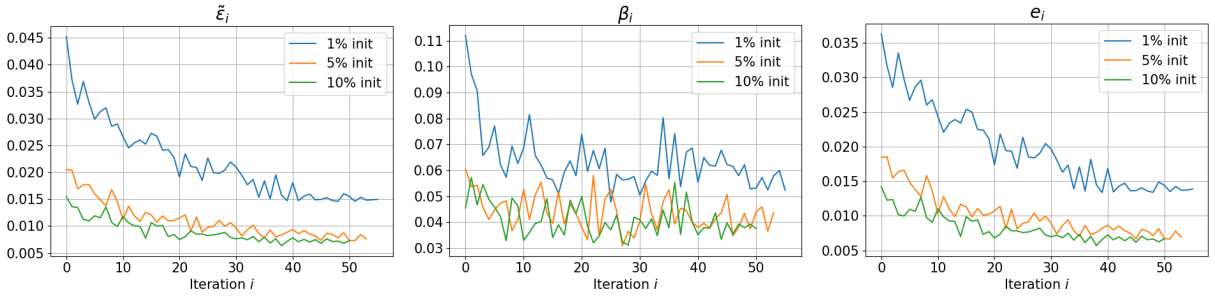


Figure 5: Average number of user annotations/interactions that MISP-NEIL requests for each user question during iterative training (on WikiSQL), when the parser is initialized using 10%, 5% and 1% of training data.



(a) (b) (c)

Figure 6: The values of ~$_i$ (a), $_i$ (b) and e$_i$ (c) in MISP-NEIL throughout the training (on WikiSQL validation set), under different initialization settings.