Context-Dependent Semantic Parsing over Temporally Structured Data

Charles Chen and Razvan Bunescu
School of Electrical Engineering and Computer Science, Ohio University
lc971015@ohio.edu, bunescu@ohio.edu

Abstract

We describe a new semantic parsing setting that allows users to query the system using both natural language questions and actions within a graphical user interface. Multiple time series belonging to an entity of interest are stored in a database and the user interacts with the system to obtain a better understanding of the entity's state and behavior, entailing sequences of actions and questions whose answers may depend on previous factual or navigational interactions. We design an LSTM-based encoder-decoder architecture that models context dependency through copying mechanisms and multiple levels of attention over inputs and previous outputs. When trained to predict tokens using supervised learning, the proposed architecture substantially outperforms standard sequence generation baselines. Training the architecture using policy gradient leads to further improvements in performance, reaching a sequence-level accuracy of 88.7% on artificial data and 74.8% on real data.

1 Introduction and Motivation

Wearable sensors are being increasingly used in medicine to monitor important physiological parameters. Patients with type I diabetes, for example, wear a sensor inserted under the skin which provides measurements of the interstitial blood glucose level (BGL) every 5 minutes. Sensor bands provide a non-invasive solution to measuring additional physiological parameters, such as temperature, skin conductivity, heart rate, and acceleration of body movements. Patients may also self-report information about discrete life events such as meals, sleep, or stressful events, while an insulin pump automatically records two types of insulin interventions: a continuous stream of insulin called the basal rate, and discrete self-administered insulin dosages called boluses. The data acquired from sensors and patients accumulates rapidly and leads to a substantial data overload for the health provider.

To help doctors more easily browse the wealth of generated patient data, we built a graphical user interface (GUI) that displays the various time series of measurements corresponding to a patient. As shown in Figure 1, the GUI displays the data corresponding to one day, whereas buttons allow the user to move to the next or previous day. While the graphical interface was enthusiastically received by doctors, it soon became apparent that the doctor-GUI interaction could be improved substantially if the tool also allowed for natural language (NL) interactions. Most information needs are highly contextual and local. For example, if the blood glucose spiked after a meal, the doctor would often want to know more details about the meal or about the bolus that preceded the meal. The doctor often found it easier to express their queries in natural language (e.g. "show me how much he ate", "did he bolus before that"), resulting in a sub-optimal situation where the doctor would ask this type of *local questions* in English while a member of our team would perform the clicks required to answer the question, e.g. click on the meal event, to show details such as amount of carbohydrates. Furthermore, there were also *global questions*, such as "How often does the patient go low in the morning and the evening", whose answers would require browsing the entire patient history in the worst case, which would be very inefficient. This motivated us to start work on a new system component that would allow the doctor to interact using both natural language queries and direct actions within the GUI. A successful solution to the task described in this paper has the potential for applications in many areas of medicine where sensor data and life events are pervasive. Intelligent user interfaces for the proposed task will also benefit the exploration and interpretation of data in other domains such as experimental

Proceedings of NAACL-HLT 2019, pages 3576–3585
Minneapolis, Minnesota, June 2 - June 7, 2019.

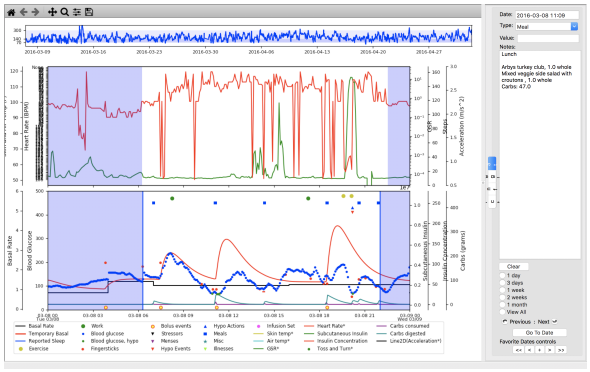

Figure 1: GUI window displaying 1 day worth of data.

physics, where large amounts of time series data are generated from high-throughput experiments.

2 Task Definition

Given an input from the user (a NL query or a direct GUI interaction), the aim is to parse it into a logical form representation that can be run by an inference engine in order to automatically extract the answer from the database. Table 1 shows sample inputs paired with their logical forms. For each input, the examples also show relevant previous inputs from the interaction sequence. In the following sections we describe a number of major features that, on their own or through their combination, distinguish this task from other semantic parsing tasks.

2.1 Time is essential

All events and measurements in the knowledge base are organized in time series. Consequently, many queries contain time expressions, such as the relative “midnight" or the coreferential “then", and temporal relations between relevant entities, expressed through words such as “after" or “when". This makes processing of temporal relations essential for a good performance. Furthermore, the GUI serves to anchor the system in time, as most of the information needs expressed in local questions are relative to the day shown in the GUI, or the last event that was clicked.

2.2 GUI interactions vs. NL questions

The user can interact with the system 1) directly within the GUI (e.g. mouse clicks); 2) through natural language questions; or 3) through a combination of both, as shown in Examples 1 and 2 in Table 1. Although the result of every direct interaction with the GUI can also be obtained using natural language questions, sometimes it can be more convenient to use the GUI directly, especially when all events of interest are in the same area of the screen and thus easy to move the mouse or hand from one to the other. For example, a doctor interested in what the patient ate that day can simply click on the blue squares at the top of the bottom pane in Figure 1, one after another. Sometimes a click can be used to anchor the system at a particular time during the day, after which the doctor can ask short questions implicitly focused on that region in time. An example of such hybrid behavior is shown in Example 2, where a click on a Bolus event is followed by a question about a snack, which implicitly should be the meal right after the bolus.

Example 1
Click on Exercise event at 9:29am. $Click(e) \wedge e.type = \text{Exercise} \wedge e.time = \text{9:29am}$
Click on Miscellaneous event at 9:50am $Click(e) \wedge e.type = \text{Misc} \wedge e.time = \text{9:50am}$
Q_1: What was she doing mid afternoon when her heart rate went up? $Answer(e) \wedge Behavior(e_1.value, \text{Up})$ $\wedge Around(e.time, e_1.time)$ $\wedge e.type == DiscreteType$ $\wedge e1.type == HeartRate$ $\wedge e1.time == MidAfternoon()$
Q_2: What time did that start? $Answer(e(-1).time)$
Example 2
Click on Bolus at 8:03pm. $Click(e) \wedge e.type = \text{Bolus} \wedge e.time = \text{8:03pm}$
Q_3: What did she eat for her snack? $Answer(e.food) \wedge e.kind == Snack$
Example 3
Click on Exercise at 7:52pm. $Click(e) \wedge e.type = \text{Exercise} \wedge e.time = \text{7:52pm}$
Q_4: What did she do then? $Answer(e(-1).kind)$
Q_5: Did she take a bolus before then? $Answer(Any(d.type == Bolus$ $\wedge Before(d.time, e(-1).time)))$
Example 4
Q_6: What is the first day they have heart rate reported? $Answer(e.date)$ $\wedge Order(e, 1, Sequence(d, d.type == HeartRate))$
Example 5
Q_7: Is there another day he goes low in the morning? $Answer(Any(Hypo(d1) \wedge x! = CurrentDate$ $\wedge x.type == Date \wedge d1.time == Morning(x))$

Table 1: Examples of interactions and logical forms.

2.3 Factual queries vs. GUI commands

Most of the time, doctors have information needs that can be satisfied by clicking on an event shown in the GUI or by asking factual questions about a particular event of interest from that day. In con-

trast, a different kind of interaction happens when the doctor wants to change what is shown in the tool, such as toggling on/off particular time series (e.g. “toggle on heart rate"), or navigating to a different day (e.g. “go to next day", “look at the previous day"). Sometimes, a question may be a combination of both, as in “What is the first day they have a meal without a bolus?", for which the expectation is that the system navigates to that day and also clicks on the meal event to show additional information and anchor the system at the time of that meal.

2.4 Sequential dependencies

The user interacts with the system through a sequence of questions or clicks. The logical form of a question, and implicitly its answer, may depend on the previous interaction with the system. Examples 1 to 3 in Table 1 are all of this kind. In example 1, the pronoun “that" in question 2 refers to the answer to question 1. In example 2, the snack refers to the meal around the time of the bolus event that was clicked previously – this is important, as there may be multiple snacks that day. In example 3, the adverb “then" in question 5 refers to the time of the event that is the answer of the previous question. As can be seen from these examples, sequential dependencies can be expressed as coreference between events from different questions. Coreference may also happen within questions, as in question 4 for example. Overall, solving coreferential relations will be essential for good performance.

3 Semantic Parsing Datasets

To train and evaluate semantic parsing approaches, we created two datasets of sequential interactions: a dataset of real interactions (Section 3.1) and a much larger dataset of artificial interactions (Section 3.2).

3.1 Real Interactions

We recorded interactions with the GUI in real time, using data from 9 patients, each with around 8 weeks worth of time series data. In each recording session, the tool was loaded with data from one patient and the physician was instructed to explore the data in order to understand the patient behavior as usual, by asking NL questions or interacting directly with the GUI. Whenever a question was asked, a member of our study team found the answer by navigating in and clicking on the corresponding event. After each session, the question segments were extracted manually from the speech recordings, transcribed, and timestamped. All direct interactions (e.g. mouse clicks) were recorded automatically by the tool, timestamped, and exported into an XML file. The sorted list of questions and the sorted list of mouse clicks were then merged using the timestamps as key, resulting in a chronologically sorted list of questions and GUI interactions. Mouse clicks were automatically translated into logical forms, whereas questions were parsed into logical forms manually.

Event Types
Physiological Parameters:
BGL, BasalRate, TemporaryBasal, Carbs, GSR, InfusionSet, AirTemperature, SkinTemperature, HeartRate, StepCount.
Life Events:
FingerSticks, Bolus, Hypo, HypoAction, Misc, Illness, Meal, Exercise, ReportedSleep, Wakeup, Work, Stressors.
Constants
Up, Down, On, Off, Monday, Tuesday, ..., Sunday.
Functions
$Interval(t_1, t_2)$, $Before(t)$, $After(t)$, ...
return corresponding intervals (default lengths).
$Morning([d])$, $Afternoon([d])$, $Evening([d])$, ...
return corresponding intervals for day d.
$WeekDay(d)$:
return the day of the week of date d.
$Sequence(var, statements)$:
return a chronologically ordered sequence of possible values for var that satisfy $statements$.
$Count(var[, statements])$:
returns the number of possible values for var that satisfy $statements$.
Predicates
$Answer(e)$, $Click(e)$
$Morning(t)$, $Afternoon(t)$, $Evening(t)$, ...
$Overlap(t_1, t_2)$, $Before(t_1, t_2)$, $Around(t_1, t_2)$, ...
$Behavior(variable, direction)$:
whether $variable$ increases, if $direction$ is Up, (or decrease if $direction$ is $Down$).
$High(variable)$, $Low(variable)$:
whether $variable$ has some low value.
$Order(event, ordinal, sequence[, attribute])$:
whether the $event$ is at place $ordinal$ in $sequence$
Commands
DoClick, DoToggle, DoSetDate, DoSetTime, ...

Table 2: Vocabulary for logical forms.

A snapshot of the vocabulary for logical forms is shown in Table 2, showing the Event Types, Constants, Functions, Predicates, and Commands. Every life event or physiological measurement stored in the database is represented in the logical forms as an event object e with 3 major attributes: $e.type$, $e.date$, and $e.time$. Depending on its type, an event object may contain additional fields. For example, if $e.type = BGL$, then it has an attribute $e.value$. If $e.type = Meal$, then it has attributes $e.food$ and $e.carbs$. We use $e(-i)$ to represent the event

appearing in the i^{th} previous logical form (LF). Thus, to reference the event mentioned in the previous LF, we use $e(-1)$, as shown for question Q_5. If more than one event appears in the previous LF, we use an additional index j to match the event index in the previous LF. Coreference between events is represented simply using the equality operator, e.g. $e = e(-1)$.The dataset contains logical forms for 237 interactions: 74 mouse clicks and 163 NL queries.

3.2 Artificial Interactions

The number of annotated real interactions is too small for training an effective semantic parsing model. To increase the number of training examples, we designed and implemented an artificial data generator that simulates user-GUI interactions, with sentence *templates* defining the skeleton of each entry in order to maintain high-quality sentence structure and grammar. This approach is similar to (Weston et al., 2015), with the difference that we need a much higher degree of variation such that the machine learning model does not memorize all possible sentences, and consequently a much richer template database. We therefore implemented a *template language* with recursive grammar, that can be used to define as many templates and generate as many data examples as desired. We used the same vocabulary as for the real interactions dataset. To generate contextual dependencies (e.g. event coreference), the implementation allows for more complex *combo templates* where a sequence of templates are instantiated together. A more detailed description of the template language and the simulator implementation is given in (Chen et al., 2019) and Appendix A, together with illustrative examples. The simulator was used to generate 1,000 interactions and their logical forms: 312 mouse clicks and 688 NL queries.

4 Baseline Models for Semantic Parsing

This section describes two baseline models: a standard LSTM encoder-decoder for sequence generation *SeqGen* (Section 4.1) and its attention-augmented version *SeqGen+Att2In* (Section 4.2). This last model will be used later in Section 5 as a component in the context-dependent semantic parsing architecture.

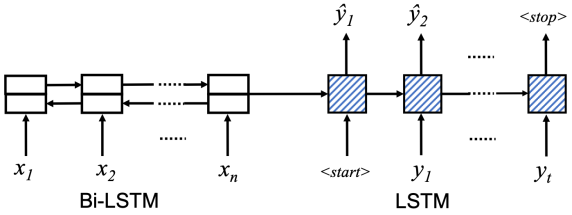

Figure 2: The *SeqGen* model takes a sequence of interactions as input $X = x_1, \ldots, x_n$ and encodes it with a Bi-LSTM (left). The decoder LSTM (right) generates a logical form $\hat{Y} = \hat{y}_1, \ldots, \hat{y}_T$.

4.1 *SeqGen*

As shown in Figure 2, the sequence-generation model *SeqGen* uses Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) units in an encoder-decoder architecture (Bahdanau et al., 2017; Cho et al., 2014), composed of a bi-directional LSTM for the encoder over the input sequence X and an LSTM for the decoder of the output LF sequence Y. We use $Y_t = y_1, \ldots, y_t$ to denote the sequence of output tokens up to position t. We use $\hat{Y}$ to denote the generated logical form.

The initial state $\boldsymbol{s}_0$ is created by running the bi-LSTM encoder over the input sequence X and concatenating the last hidden states. Starting from the initial hidden state $\boldsymbol{s}_0$, the decoder produces a sequence of states $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T$, using embeddings $e(y_t)$ to represent the previous tokens in the sequence. A softmax is used to compute token probabilities at each position as follows:

$$p(y_t|Y_{t-1}, X) = softmax(\boldsymbol{W}_h \boldsymbol{s}_t) \qquad (1)$$
$$\boldsymbol{s}_t = h(\boldsymbol{s}_{t-1}, e(y_{t-1}))$$

The transition function h is implemented by the LSTM unit.

4.2 *SeqGen+Att2In*

This model (Figure 3) is similar to *SeqGen*, except that it attends to the current input (NL query or mouse click) during decoding. Equation 2 defines the corresponding attention mechanism *Att2In* used to create the context vector $\boldsymbol{d}_t$:

$$e_{tj} = \boldsymbol{v}_a^T \tanh(\boldsymbol{W}_a \boldsymbol{f}_j + \boldsymbol{U}_a \boldsymbol{s}_{t-1}) \qquad (2)$$
$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{m} \exp(e_{tk})}, \quad \boldsymbol{d}_t = \boldsymbol{c}_t = \sum_{j=1}^{n} \alpha_{tj} \boldsymbol{f}_j$$

Here $\boldsymbol{f}_j$ is the j-th hidden states for Bi-LSTM corresponding to $\boldsymbol{x}_j$ and α_{tj} is an attention weight.

Both the context vector $\boldsymbol{d}_t$ and $\boldsymbol{s}_t$ are used to predict the next token $\hat{y}_t$ in the logical form:

$$\hat{y}_t \sim \text{softmax}(\boldsymbol{W}_h \boldsymbol{s}_t + \boldsymbol{W}_d \boldsymbol{d}_t)$$

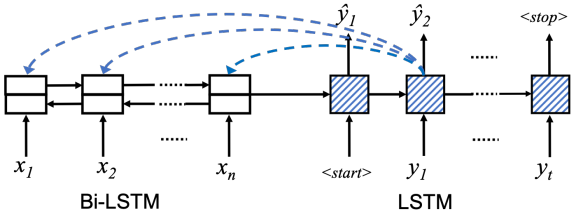

Figure 3: The *SeqGen+Att2In* model augments the *SeqGen* model with an attention mechanism. At each decoding step t, it attends to all input tokens in order to compute a context vector $\boldsymbol{d}_t$.

5 Context-Dependent Semantic Parsing

In Figure 4 we show our proposed semantic parsing model, *SP+Att2All+Copy* (*SPAAC*). Similar to the baseline models, we use a bi-directional LSTM to encode the input and another LSTM as the decoder. Context-dependency is modeled using two types of mechanisms: *attention* and *copying*. The attention mechanism (Section 5.1) is comprised of 3 models: *Att2HisIn* attending to the previous input, *Att2HisLF* attending to the previous logical form, and the *Att2In* introduced in Section 4.2 that attends to the current input. The copying mechanism (Section 5.2) is comprised of two models: one for handling unseen tokens, and one for handling coreference to events in the current and previous logical forms.

5.1 Attention Mechanisms

At decoding step t, the *Att2HisIn* attention model computes the context vector $\hat{\boldsymbol{c}}_t$ as follows:

$$\hat{e}_{tk} = \boldsymbol{v}_b^T \tanh(\boldsymbol{W}_b \boldsymbol{r}_k + \boldsymbol{U}_b \boldsymbol{s}_{t-1}) \quad (3)$$

$$\beta_{tk} = \frac{\exp(\hat{e}_{tk})}{\sum_{l=1}^{m^2} \exp(\hat{e}_{tl})}, \quad \hat{\boldsymbol{c}}_t = \sum_{k=1}^{n} \beta_{tk} \cdot \boldsymbol{r}_k$$

where $\boldsymbol{r}_k$ is the encoder hidden state corresponding to $\boldsymbol{x}_k$ in the previous input X^{-1}, $\hat{\boldsymbol{c}}_t$ is the context vector, and β_{tk} is an attention weight.

Similarly, the *Att2HisLF* model computes the context vector $\tilde{\boldsymbol{c}}_t$ as follows:

$$\tilde{e}_{tj} = \boldsymbol{v}_c{}^T \tanh(\boldsymbol{W}_c \boldsymbol{l}_j + \boldsymbol{U}_c \boldsymbol{s}_{t-1}) \quad (4)$$

$$\gamma_{tj} = \frac{\exp(\tilde{e}_{tj})}{\sum_{j=1}^{n} \exp(\tilde{e}_{tj})}, \quad \tilde{\boldsymbol{c}}_t = \sum_{j=1}^{n} \gamma_{tj} \cdot \boldsymbol{l}_j$$

where $\boldsymbol{l}_j$ is the j-th hidden state of the decoder for the previous logical form Y^{-1}.

The context vector used in the decoder is comprised of the context vectors from the three attention models *Att2In*, *Att2HisIn* and *Att2HisLF*:

$$\boldsymbol{d}_t = concat(\boldsymbol{c}_t, \hat{\boldsymbol{c}}_t, \tilde{\boldsymbol{c}}_t) \quad (5)$$

5.2 Copying Mechanisms

In order to handle out-of-vocabulary (OOV) tokens and coreference (REF) between entities in the current and the previous logical forms, we add two special tokens *OOV* and *REF* to the vocabulary. Inspired by the copying mechanism in (Gu et al., 2016), we train the model to learn which token in the current input $X = \{x_j\}$ is an OOV by minimizing the following loss:

$$L_{oov}(Y) = -\sum_{t=1}^{Y.l} \sum_{j=1}^{X.l} \log p_o(O_j | \boldsymbol{s}_j^X, \boldsymbol{s}_t^Y) \quad (6)$$

where $X.l$ is the length of current input, $Y.l$ is the length of the current logical form, $\boldsymbol{s}_j^X$ is the LSTM state for x_j and $\boldsymbol{s}_t^Y$ is the LSTM state for y_t, $O_j \in \{0, 1\}$ is a label indicating whether x_j is an OOV. We use logistic regression to compute the OOV probability, i.e. $p_o(O_j = 1 | \boldsymbol{s}_j^X, \boldsymbol{s}_t^Y) = \sigma(\boldsymbol{w}_o^T [\boldsymbol{s}_j^X, \boldsymbol{s}_t^Y])$.

Similarly, to solve coreference, the model is trained to learn which entity in the previously generated logical form $\hat{Y}^{-1} = \{\hat{y}_j\}$ is coreferent with the entity in the current logical form by minimizing the following loss:

$$L_{ref}(Y) = -\sum_{t=1}^{Y.l} \sum_{j=1}^{\hat{Y}^{-1}.l} \log p_r(R_j | \boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) \quad (7)$$

where $\hat{Y}^{-1}.l$ is the length of the previous generated logical form, $Y.l$ is the length of the current logical form, $\boldsymbol{s}_j^{\hat{Y}^{-1}}$ is the LSTM state at position j in $\hat{Y}^{-1}$ and $\boldsymbol{s}_t^Y$ is the LSTM state for position t in Y, and $R_j \in \{0, 1\}$ is a label indicating whether $\hat{y}_j$ is an entity referred by y_t in the next logical form Y. We use logistic regression to compute the coreference probability, i.e. $p_r(R_j = 1 | \boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) = \sigma(\boldsymbol{w}_r^T [\boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y])$.

Finally, we use "Teacher forcing" (Williams and Zipser, 1989) to train the model to learn which token in the vocabulary (including special tokens *OOV* and *REF*) should be generated, by minimizing

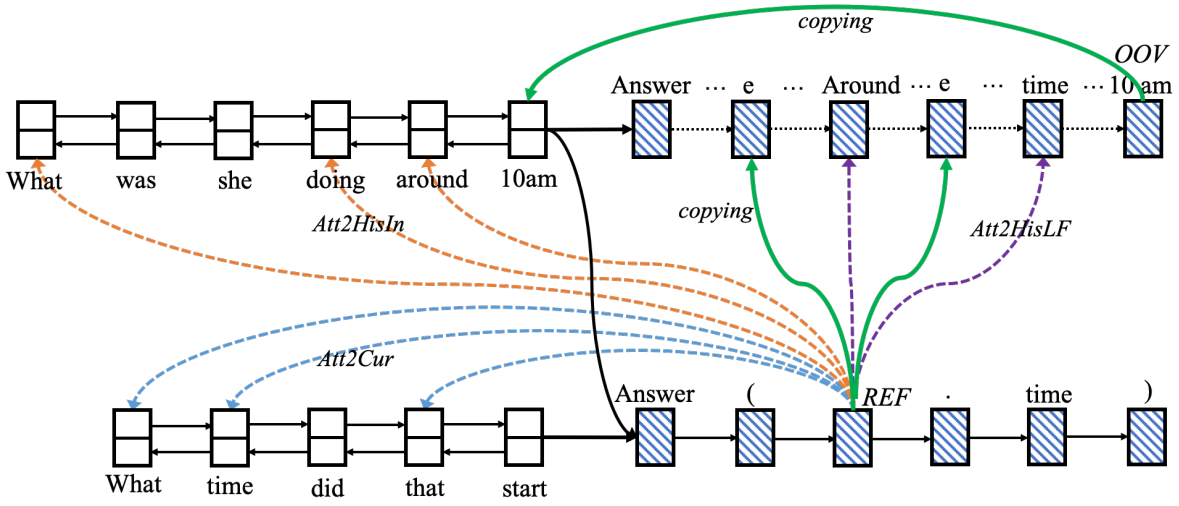

Figure 4: Context-dependent semantic parsing architecture. We use a Bi-LSTM (left) to encode the input and a LSTM (right) as the decoder. We show only parts of the LF to save space. The complete generated LF at time T-1 is Y^{-1} = [*Answer, (, e,), ∧, Around, (, e, ., time, OOV,), ∧, e, ., type, ==, DiscreteType*]. The token *10am* is copied from the input to replace the generated *OOV* token (solid green arrow). The complete generated LF at time T is Y = [*Answer, (, REF, ., time,)*]. The entity token *e* is copied from the previous LF to replace the generated *REF* token (solid green arrow). Orange dash arrows attend to historical input. Blue dash arrows attend to current input. Purple dash arrows attend to previous logical form.

the following token generation loss:

$$L_{gen}(Y) = -\sum_{t=1}^{Y.l} \log p(y_t|Y_{t-1}, X) \quad (8)$$

where $Y.l$ is the length of the current logical form.

5.3 Supervised Learning: *SPAAC-MLE*

The supervised learning model *SPAAC-MLE* is obtained by training the semantic parsing architecture from Figure 4 to minimize the sum of the 3 negative log-likelihood losses:

$$L_{MLE}(Y) = L_{gen}(Y) + L_{oov}(Y) + L_{ref}(Y) \quad (9)$$

At inference time, beam search is used to generate the LF sequence (Ranzato et al., 2015; Wiseman and Rush, 2016). During inference, if the generated token at position t is *OOV*, we copy the token from the current input X that has the maximum OOV probability, i.e. $\arg\max_j p_o(O_j = 1|\boldsymbol{s}_j^X, \boldsymbol{s}_t^Y)$. Similarly, if the generated entity token at position t is *REF*, we copy the entity token from the previous LF Y^{-1} that has the maximum coreference probability, i.e. $\arg\max_j p_r(R_j = 1|\boldsymbol{s}_j^{Y^{-1}}, \boldsymbol{s}_t^Y)$.

5.4 Reinforcement Learning: *SPAAC-RL*

All models described in this paper are evaluated using sequence-level accuracy, a discrete metric where a generated logical form is considered to be correct if it is equivalent with the ground truth logical form. This is a strict evaluation measure in the sense that it is sufficient for a token to be wrong to invalidate the entire sequence. At the same time, there can be many generated sequences that are correct, e.g. any reordering of the clauses from the ground truth sequence is correct. The large number of potentially correct generations can lead MLE-trained models to have sub-optimal performance (Paulus et al., 2017; Rennie et al., 2017; Zeng et al., 2016; Norouzi et al., 2016). Furthermore, although "teacher forcing" (Williams and Zipser, 1989) is widely used for training sequence generation models, it leads to *exposure bias* (Ranzato et al., 2015): the network has knowledge of the ground truth LF tokens up to the current token during training, but not during testing, which can lead to propagation of errors at generation time.

Like Paulus et al. (2017), we address these problems by using policy gradient to train a token generation policy that aims to directly maximize sequence-level accuracy. We use the self-critical policy gradient training algorithm proposed by Rennie et al. (2017). We model the sequence generation process as a sequence of actions taken according to a policy, which takes an action (token $\hat{y}_t$) at each step t as a function of the current state (history $\hat{Y}_{t-1}$), according to the probability $p(\hat{y}_t|\hat{Y}_{t-1})$. The algorithm uses this probability to define two policies: a greedy, baseline policy π^b

that takes the action with the largest probability, i.e. $\pi^b(\hat{Y}_{t-1}) = \arg\max_{\hat{y}_t} p(\hat{y}_t|\hat{Y}_{t-1})$; and a sampling policy π^s that samples the action according to the same distribution, i.e. $\pi^s(\hat{Y}_{t-1}) \propto p(\hat{y}_t|\hat{Y}_{t-1})$.

The baseline policy is used to generate a sequence $\hat{Y}^b$, whereas the sampling policy is used to generate another sequence $\hat{Y}^s$. The reward $R(\hat{Y}^s)$ is then defined as the difference between the sequence-level accuracy (A) of the sampled sequence $\hat{Y}^s$ and the baseline sequence $\hat{Y}^b$. The corresponding self-critical policy gradient loss is:

$$\begin{aligned} L_{RL} &= -R(\hat{Y}^s) \times L_{MLE}(\hat{Y}^s) \\ &= -\Big(A(\hat{Y}^s) - A(\hat{Y}^b)\Big) \times L_{MLE}(\hat{Y}^s) \quad (10) \end{aligned}$$

Thus, minimizing the RL loss is equivalent to maximizing the likelihood of the sampled $\hat{Y}^s$ if it obtains a higher sequence-level accuracy than the baseline $\hat{Y}^b$.

6 Experimental Evaluation

All models are implemented in Tensorflow using dropout to deal with overfitting. For both datasets, 10% of the data is put aside for validation. After tuning on the artificial validation data, the feed-forward neural networks dropout rate was set to 0.5 and the LSTM units dropout rate was set to 0.3. The word embeddings had dimensionality of 64 and were initialized at random. Optimization is performed with the Adam algorithm. For each dataset, we use five-fold cross evaluation, where the data is partitioned into five folds, one fold is used for testing and the other folds for training. The process is repeated five times to obtain test results on all folds. We use an early-stop strategy on the validation set. The number of gradient updates is typically more than 20,000. All the experiments are performed on a single NVIDIA GTX1080 GPU.

The models are trained and evaluated on the artificial interactions first. To evaluate on real interactions, the models are pre-trained on the entire artificial dataset and then fine-tuned using real interactions. *SPAAC-RL* is pre-trained with MLE loss to provide more efficient policy exploration. We use sequence level accuracy as evaluation metric for all models: a generated sequence is considered correct if and only if all the generated tokens match the ground truth tokens.

We report experimental evaluations of the proposed models *SPAAC-MLE* and *SPAAC-RL* and baseline models *SeqGen*, *SeqGen+Att2In* on the Real and Artificial Interactions Datasets in Table 3. We also report examples generated by the SPAAC models in Tables 4 and 5.

Models	Artificial	Real
SeqGen	51.8	22.2
SeqGen+Att2In	72.7	35.4
SPAAC-MLE	**84.3**	**66.9**
SPAAC-RL	**88.7**	**74.8**

Table 3: Sequence-level accuracy on the 2 datasets.

Well the Finger Stick is 56. T&MLE&RL: $e.type == Fingerstick \wedge e.value == 56$ It looks like she suspended her pump. T&MLE&RL: $Suspended(e) \wedge around(e.time, e(-1).time)$
Let's look at the next day. T&MLE&RL: $DoSetDate(currentdate + 1)$
See if he went low. T&MLE&RL: $Answer(any(e, hypo(e)))$
Let's see what kind of exercise that is, where the steps are high? T&RL:$Answer(e.kind) \wedge e.type == exercise$ $\wedge around(e.time, e_1.time) \wedge e_1.type == \boldsymbol{stepcount}$ $\wedge \boldsymbol{high(e_1.value)}$ MLE: $Answer(e.kind) \wedge e.type == exercise$ $\wedge around(e.time, e_1.time) \wedge e_1.type == \boldsymbol{exercise}$ $\wedge \boldsymbol{e_1.type == exercise}$
Click on the exercise. T&RL:$\boldsymbol{DoClick}(e) \wedge e.type == exercise$ MLE: $\boldsymbol{Answer}(e) \wedge e.type == exercise$

Table 4: Examples generated by *SPAAC-MLE* and *SPAAC-RL* using real interactions. T: true logical forms. MLE: logical forms by *SPAAC-MLE*. RL: logical forms by *SPAAC-RL*.

6.1 Discussion

The results in Table 3 demonstrate the importance of modeling context-dependency, as the two *SPAAC* models outperform the baselines on both datasets. The RL model also obtains substantially better accuracy than the MLE model. The improvement in performance over the MLE model for the real data is statistically significant at $p = 0.05$ in a one-tailed paired t-test.

Analysis of the generated logical forms revealed that one common error made by *SPAAC-MLE* is the generation of incorrect event types. Some of these errors are fixed by the current RL model. However, there are instances where even the RL-trained model outputs the wrong event type. By comparing

Does he always get some sleep around 4:30pm? T&MLE&RL: $Answer(cond(around(x, 4:30pm)$ $=> any(e.type == reportedsleep \wedge e.time == x)))$
Is it the first week of the patient? T&MLE&RL: $Answer(week(currentdate) == x)$ $\wedge order(x, 1, sequence(e, e.type == week))$
Does she ever get some rest around 5:37pm? T&MLE&RL: $Answer(any(e.type == reportedsleep$ $\wedge around(e.time, 5:37pm)))$
When is the first time he changes his infusion set? T&MLE&RL: $Answer(e.date)$ $\wedge order(e, 1, sequence(e, e.type == infusionset))$
How many months she has multiple exercises? T&RL: $Answer(count(x, count(e, e.type == exercise$ $\wedge e.date == x) > 1 \wedge x.type == \boldsymbol{month}))$ MLE: $Answer(count(x, count(e, e.type == exercise$ $\wedge e.date == x) > 1 \wedge x.type == \boldsymbol{week}))$
Toggle so we can see fingersticks. T&RL: $DoToggle(on, \boldsymbol{fingersticks})$ MLE: $DoToggle(on, \boldsymbol{bgl})$

Table 5: Examples generated by *SPAAC-MLE* and *SPAAC-RL* using artificial interactions. T: true logical forms. MLE: logical forms generated by *SPAAC-MLE*. RL: logical forms generated by *SPAAC-RL*.

the sampled logical forms $\hat{Y}^s$ and the generated baseline logical forms $\hat{Y}^b$, we found that sometimes the sampled tokens for event types are the same as those in the baseline. An approach that we plan to investigate in future work is to utilize more advanced sampling methods to generate $\hat{Y}^s$, in order to achieve a better balance between exploration and exploitation.

7 Related Work

Question Answering has been the topic of recent research (Yih et al., 2014; Dong et al., 2015; Andreas et al., 2016; Hao et al., 2017; Abujabal et al., 2017; Chen and Bunescu, 2017). Semantic parsing, which maps text in natural language to meaning representations in formal logic, has emerged as an important component for building QA systems, as in (Liang, 2016; Jia and Liang, 2016a; Zhong et al., 2017). Context-dependent processing has been explored in complex, interactive QA (Harabagiu et al., 2005; Kelly and Lin, 2007) and semantic parsing (Zettlemoyer and Collins, 2009; Artzi and Zettlemoyer, 2011; Iyyer et al., 2017; Suhr et al., 2018; Long et al., 2016). Although these approaches take into account sequential dependencies between questions or sentences, the setting in our work has a number of significant distinguishing features, such as the importance of time – data is represented naturally as multiple time series of events – and the anchoring on a graphical user interface that also enables direct interactions through mouse clicks and a combination of factual queries and interface commands.

Dong and Lapata (2016) use an attention-enhanced encoder-decoder architecture to learn the logical forms from natural language without using hand-engineered features. Their proposed Seq2Tree architecture can capture the hierarchical structure of logical forms. Jia and Liang (2016b) train a sequence-to-sequence RNN model with a novel attention-based copying mechanism to learn the logical forms from questions. The copying mechanism has been investigated by Gu et al. (2016) and Gulcehre et al. (2016) in the context of a wide range of NLP applications. These semantic parsing models considered sentences in isolation. In contrast, generating correct logical forms in our task required modeling sequential dependencies between logical forms. In particular, coreference is modeled between events mentioned in different logical forms by repurposing the copying mechanism originally used for modeling out-of-vocabulary tokens.

8 Conclusion

We introduced a new semantic parsing setting in which users can query a system using both natural language and direct interactions (mouse clicks) within a graphical user interface. Correspondingly, we created a dataset of real interactions and a much larger dataset of artificial interactions. The correct interpretation of a natural language query often requires knowledge of previous interactions with the system. We proposed a new sequence generation architecture that modeled this context dependency through multiple attention models and a copying mechanism for solving coreference. The proposed architecture is shown to outperform standard LSTM encoder-decoder architectures that are context agnostic. Furthermore, casting the sequence generation process in the framework of reinforcement learning alleviates the exposure bias and leads to substantial improvements in sequence-level accuracy.

The two datasets and the implementation of the systems presented in this paper are made publicly available at `https://github.com/charleschen1015/SemanticParsing`. The data visualization GUI is available under

the name OHIOT1DMVIEWER at `http://smarthealth.cs.ohio.edu/nih.html`.

Acknowledgments

This work was partly supported by grant 1R21EB022356 from the National Institutes of Health. We would like to thank Frank Schwartz and Cindy Marling for contributing real interactions, Quintin Fettes and Yi Yu for their help with recording and pre-processing the interactions, and Sadegh Mirshekarian for the design of the artificial data generation. We would also like to thank the anonymous reviewers for their useful comments.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2017. Quint: Interpretable question answering over knowledge bases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–66.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, pages 1545–1554.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the conference on empirical methods in natural language processing*, pages 421–432. Association for Computational Linguistics.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Charles Chen and Razvan Bunescu. 2017. An exploration of data augmentation and rnn architectures for question ranking in community question answering. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 442–447.

Charles Chen, Sadegh Mirshekarian, Razvan Bunescu, and Cindy Marling. 2019. From physician queries to logical forms for efficient exploration of patient data. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 371–374. IEEE.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 33–43.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. 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)*, volume 1, pages 260–269.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 140–149.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 221–231.

Sanda Harabagiu, Andrew Hickl, John Lehmann, and Dan Moldovan. 2005. Experiments with interactive question-answering. In *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*, pages 205–214. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1821–1831.

Robin Jia and Percy Liang. 2016a. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016b. Data recombination for neural semantic parsing. In *Proceedings of*

the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 12–22.

Diane Kelly and Jimmy Lin. 2007. Overview of the TREC 2006 ciQA task. In *ACM SIGIR Forum*, volume 41, pages 107–116. ACM.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1456–1465.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195. IEEE.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2238–2249.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 643–648.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.