2019 IEEE 13th International Conference on Semantic Computing (ICSC)

From Physician Queries to Logical Forms for Efficient Exploration of Patient Data

Charles Chen, Sadegh Mirshekarian, Razvan Bunescu and Cindy Marling
School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701, USA

***Abstract*—We introduce a new question answering paradigm in which users can interact with the system using natural language questions or direct actions within a graphical user interface (GUI). The system displays multiple time series characterizing the behavior of a patient, and a physician interacts with the system through GUI actions and questions, where answers may depend on previous interactions. To find the answers automatically, we propose parsing the questions into logical forms for execution by an inference engine over the underlying database. The semantic parser is implemented as an LSTM-based encoder-decoder that models dependencies between consecutive answers through multiple attention and copying mechanisms. To train and evaluate the model, we created a dataset of semantic parses of real interactions with the system, augmented with a larger dataset of artificial interactions. The proposed architecture obtains promising results, substantially outperforming standard sequence generation baselines.**

I. Introduction

Wearable sensors are being increasingly used to monitor physiological parameters that are important for the management of various medical conditions. In our work, for example, patients with type I diabetes wear a subcutaneous sensor that measures the interstitial blood glucose level (BGL) every 5 minutes. Sensor bands measure additional physiological parameters, such as temperature, skin conductivity, and heart rate. Patients may also self-report information about meals or sleep, while an insulin pump records two types of insulin delivery: a continuous stream of insulin (basal), and discrete self-administered insulin dosages (boluses). The data acquired from sensors and patients accumulates rapidly and leads to data overload for the health care provider, who may interact with the patient just a few times per year.

To help doctors more easily browse the large quantity of patient data, we built PhysioGraph, a graphical user interface (GUI) that shows a day-by-day plot of the time series of measurements acquired from a patient and offers basic browsing capabilities to navigate through the data. Doctors can click on discrete events displayed in the GUI window in order to show details such as the amount of insulin in a bolus or carbohydrates in a meal. To the best of our knowledge, PhysioGraph is the first interactive tool providing graphical visualization of aggregated patient data for diabetes specialists. While the doctors found PhysioGraph to be very useful, we soon realized that the system could be improved substantially if it also allowed doctors to ask questions in natural language (NL).

TABLE I
Examples of interactions and logical forms.

Example 1
Click on Exercise event at 9:29am. $Click(e) \wedge e.type ==$ Exercise $\wedge\, e.time ==$ 9:29am
Click on Miscellaneous event at 9:50am $Click(e) \wedge e.type ==$ Misc $\wedge\, e.time ==$ 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 $\wedge\, e_1.type ==$ HeartRate $\wedge\, e_1.time == MidAfternoon()$
Q_2: What time did that start? $Answer(e(-1).time))$
Example 2
Click on Bolus at 8:03pm. $Click(e) \wedge e.type ==$ Bolus $\wedge\, e.time ==$ 8:03pm
Q_3: What did she eat for her snack? $Answer(e.food) \wedge e.kind ==$ Snack

II. Task Definition

Given a natural language question or statement, the task is to translate its meaning into a logical form representation. Table I shows sample inputs in context, paired with their logical forms. This task presents a mix of features that distinguish it from other question answering or semantic parsing tasks. First, all events and measurements in the knowledge base are organized in time series. Therefore, queries often use time expressions and temporal relations. The GUI implicitly serves to anchor the information needs in time, as many of the queries are relative to the day shown in the GUI or the last event that was clicked. Furthermore, 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 I. 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. 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, as in Example 2, where a click on a Bolus event is followed by a question about a snack, which implicitly should be the meal immediately following the bolus.

The user interacts with PhysioGraph through a sequence of questions or clicks. The logical form of a question, and implicitly its answer, may depend on the previous interaction

TABLE II
SAMPLE OF VOCABULARY TOKENS FOR LOGICAL FORMS.

Event Types
Physiological Parameters:
BGL, BasalRate, HeartRate, TemporaryBasal, Carbs, InfusionSet...
Life Events:
Bolus, Hypo, Misc, Meal, Exercise, Wakeup, Stressors, Illness...
Constants
Up, Down, On, Off, CurrentDate, Monday, Tuesday, ..., Sunday.
Functions
$Interval(t_1, t_2)$, $Before(t)$, $Afternoon([d])$, $Morning([d])$...
Predicates
$Answer(e)$, $Click(e)$, $Low(e)$, $After(t_1, t_2)$, $Around(t_1, t_2)$...
$MidAfternoon(t)$, $Night(t)$, $Morning(t)$, $Afternoon(t)$...
$Behavior(variable, direction)$:
whether $variable$ increases(decreases), if $direction$ is Up(Down)
$Any(statements)$:
whether there is a set of $variables$ that make $statements$ true.
Commands
$DoToggle$, $DoSetDate$, $DoSetTime$, $DoClick$.

with the system. Examples 1 and 2 in Table I are both 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. As can be seen from these examples, sequential dependencies can be expressed as coreference between events from different questions.

III. Semantic Parsing Datasets

A. Real Interactions

We recorded interactions with PhysioGraph in real time, using data from 5 patients, each with around 8 weeks worth of time series data. In each recording session, PhysioGraph 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 PhysioGraph and clicking on the corresponding event. Mouse clicks were automatically translated into logical forms, whereas questions were parsed into logical forms manually.

A subset of the Event Types, Constants, Functions, Predicates, and Commands used in the vocabulary for logical forms is shown in Table II. A life event or physiological measurement stored in the database is represented as an event object e with 3 basic attributes: $e.type$, $e.date$, and $e.time$. Depending on its type, an event object may contain additional fields. For example, if $e.type == Meal$, then it has attributes $e.food$ and $e.carbs$. We use $e(-i)$ to denote the event appearing in the i^{th} previous logical form (LF). 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 then represented using the equality operator, e.g. $e == e(-1)$. Overall, the dataset contains logical forms for 163 NL questions and 74 mouse clicks.

TABLE III
EXAMPLES OF HOW ARTIFICIAL DATA ARE GENERATED.

Example types.
week_days→Monday | Tuesday | ... | Sunday
daily_intervals→morning | afternoon | evening | night
daily_intervals_logic→ Morning | Afternoon | Evening | Night
any_event→heart rate | bolus | blood glucose level
any_event_logic→HeartRate | Bolus | BGL
Example 1: a statement, involving referencing
Let's go to [week_days]. → $DoSetDate$([\$1])
Possible derivations:
• Let's go to Monday. → $DoSetDate$(Monday)
• Let's go to Tuesday. → $DoSetDate$(Tuesday)
Example 2: a combo statement capturing temporal dependence
[[let's / please / we can] / can we] turn the [any_event] off[\$1:./?]
$DoToggle$(Off, [\$2 : any_event_logic])
...and the [any_event] too.
$DoToggle$(Off, [\$1 : any_event_logic])
Possible derivations:
• please turn the bolus off. → $DoToggle$(Off, Bolus)
and the heart rate too. → $DoToggle$(Off, HeartRate)
• can we turn the blood glucose level off?
→ $DoToggle$(Off, BGL)
and the bolus too.→ $DoToggle$(Off, Bolus)
Example 3: a click, involving the special type clocktime
$Click(e) \wedge e.type ==$ [any_event_logic] $\wedge e.time ==$ [clocktime]
A possible derivation:
• $Click(e) \wedge e.type ==$ Bolus $\wedge e.time ==$ 12:36 PM
Example 4: a question, involving the special type range
is there [a/any] [valued_event] [more/less] than [range(-500,500)].
$Answer(Any(d.value$[\$3:>/<][\$4]
$\wedge d.type ==$[\$2:valued_event_logic]))
One possible derivation:
• is there any heart rate less than 250.
$Answer(Any(d.value < 250 \wedge d.type ==$ HeartRate))

B. Artificial Interactions

Training a semantic parsing model requires substantially more interactions than those recorded so far in PhysioGraph. Similar to [1], we implemented an artificial data generator that simulates interactions using sentence *templates*. To enable the generation of a virtually unlimited number of interactions, the *template language* was implemented with a context-free grammar. Below we show a simplification of three sample rules from the grammar:

$$\langle S \rangle \rightarrow \text{maximum heart rate on } \langle P \rangle \text{ today?}$$
$$\langle P \rangle \rightarrow \text{the day before } \langle P \rangle$$
$$\langle P \rangle \rightarrow \text{the Monday after}$$

A sample derivation using these rules is "maximum heart rate on the Monday after today?".

Our implementation allows for the definition of any number of types as non-terminals and an arbitrary number of templates as right hand sides of production rules for the starting symbol S. The doctor-GUI interactions can be categorized into three types: *questions*, *statements*, and *clicks*, where templates can be defined for each type, as shown in Table III. Since the sentence generator chooses each template randomly, the order of sentences in the dataset will be random. However, given the

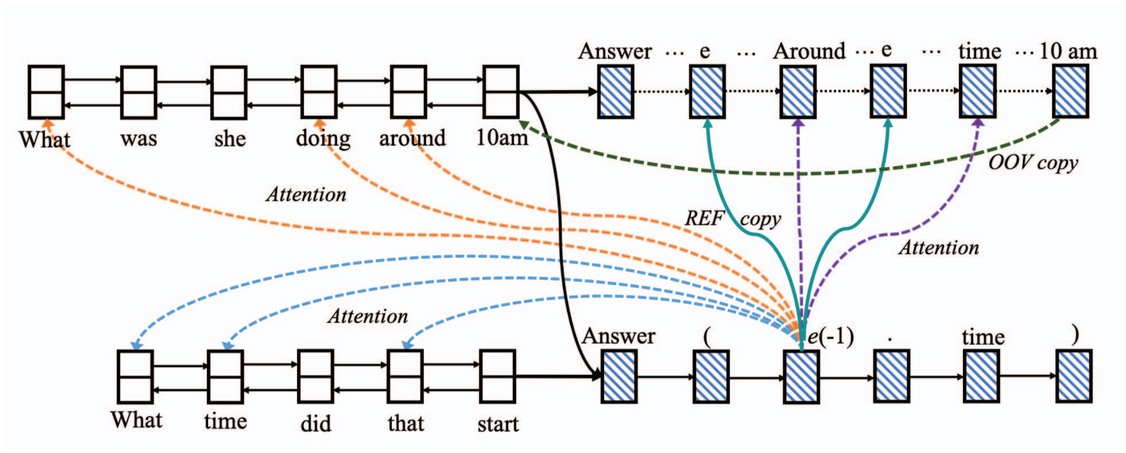

Fig. 1. Context-aware semantic parsing architecture. The complete previous generated LF is Y^{-1} = [*Answer, (, e,), ∧, Around, (, e, ., time, OOV,), ∧, e, ., type, =, DiscreteType*]. The token *10am* is copied from the input to replace the generated *OOV* token. The entity token *e* is copied from the previous LF to replace the generated *REF* token.

importance of temporal dependencies between real interactions, the implementation allows for more complex *combo* templates where multiple templates are forced to come in a given order. It is also possible to specify groups of templates, via tagging, and to combine groups rather than individual templates. Furthermore, each NL sentence template is paired with a LF template, and the two templates are instantiated jointly, using a reference mechanism to condition the logical form generation on decisions made while deriving the sentence.

Table III shows examples of how artificial sentences and their logical forms are generated given templates and types. Most types are defined using context free rules. There are however special types, such as clocktime and range, which are dynamically rewritten as a random time and a random integer from a given range, as shown in Examples 3 and 4, respectively. Note that most examples use referencing, which is a mechanism to allow for dynamic matching of terminals between the NL and LF derivations. In Example 1, $1 in the logical form template refers to the first type in the main sentence, which is week_days. This means that whatever value is substituted for week_days should appear verbatim in place of $1. In case a coordinated matching from a separate list of possible options is required, such as in Example 2, another type can be selected. In Example 2, [$2:any_event_logic] will be option i from the type any_event_logic when option i is chosen in the main sentence for the second template, which is [any_event]. We defined 82 templates and used them to generate 1,000 interactions and their logical forms, comprising 312 mouse clicks and 688 NL queries.

IV. Semantic Parsing Baselines

As baselines, we use a basic sequence-generation model with and without attention over the input tokens. The model consists of Long Short-Term Memory (LSTM) [2] units in an encoder-decoder architecture [3], 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 shown in Equation 1. In the attention version, a context vector $\boldsymbol{c}_t$ is computed for each position in the output LF, using the original model from [3].

$$\boldsymbol{s}_t = h(\boldsymbol{s}_{t-1}, e(y_{t-1}))$$
$$p(y_t|Y_{t-1}, X) = softmax(\boldsymbol{W}_h \boldsymbol{s}_t[+\boldsymbol{W}_c \boldsymbol{c}_t]) \tag{1}$$

The transition function h is implemented by the LSTM unit. To train the models, we use "teacher forcing" [4] with the following token generation loss:

$$L_{gen}(Y) = -\sum_{t=1}^{Y.l} \log p(y_t|Y_{t-1}, X) \tag{2}$$

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

V. Context-Aware Semantic Parsing

Our proposed semantic parsing model is shown in Figure 1. Similar to the baseline models, we use a bi-LSTM to encode the input and another LSTM as the decoder. Context-dependency is modeled through *attention* and a *copy* mechanism for coreference. The overall context vector $\boldsymbol{c}_t$ is the concatenation of the context vectors computed from three levels of attention: over the current input, the previous input, and the previous logical form. 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. A copying mechanism [5] is then 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:

$$L_{ref}(Y) = -\sum_{t=1}^{Y.l} \sum_{j=1}^{\hat{Y}^{-1}.l} \log p_r(R_j | \boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) \quad (3)$$

where $\boldsymbol{s}_j^{\hat{Y}^{-1}}$ 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 to by y_t in the current logical form Y. We use logistic regression to compute the coreference probability, i.e. $p_r(R_j = 1 | \boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y) = \sigma(\boldsymbol{w}_r^T [\boldsymbol{s}_j^{\hat{Y}^{-1}}, \boldsymbol{s}_t^Y])$. A similar model is used to determine which token in the current input $X = \{x_j\}$ is an OOV ($O_j \in \{0,1\}$) by minimizing the following loss:

$$L_{oov}(Y) = -\sum_{t=1}^{Y.l} \sum_{j=1}^{X.l} \log p_o(O_j | \boldsymbol{s}_j^X, \boldsymbol{s}_t^Y) \quad (4)$$

Finally, the model is trained to learn which token in the vocabulary (including special tokens *OOV* and *REF*) should be generated, by jointly minimizing the three losses:

$$L(Y) = L_{gen}(Y) + L_{oov}(Y) + L_{ref}(Y) \quad (5)$$

At inference time, beam search is used to generate the LF sequence [6]. 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 p_o. 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 p_r.

VI. Experimental Evaluation

All models are implemented in Tensorflow using dropout for regularization. For each dataset, 10% was used for tuning, 80% for training, and 10% for testing. Upon tuning, the size of word embeddings and LSTM states is set to 64, and the batch size is set to 128. Optimization is performed using Adam [7], with an initial learning rate set to 0.0001. We use an early-stop strategy on the validation set. For evaluation on real interactions, all three models are first pre-trained on the artificial dataset and then fine-tuned using the real interactions training data.

TABLE IV
Sequence-level accuracy on the two datasets.

Models	**Artificial**	**Real**
Sequence Generation	40.2	34.3
+ 1 attention level	61.6	48.6
+ 3 attention levels + OOV + REF	**73.2**	**71.4**

We use sequence level accuracy as the evaluation metric, meaning that an output logical form is considered correct if and only if all the generated tokens match the ground-truth tokens. Table IV shows the sequence-level accuracy for the two baselines and the proposed semantic parsing model. The results show that the sequence generation baselines are substantially improved by adding attention over the tokens in the input NL question. Further improvements are obtained by the full architecture, demonstrating the importance of modeling context-dependency and out-of-vocabulary tokens. Table V shows logical forms generated by the proposed full model on real NL questions. The first two are entirely correct, whereas the last one contains mistakes.

TABLE V
Examples generated on real test interactions. T indicates true logical forms and S indicates system logical forms.

Let's look at the next day. T&S:$DoSetDate$(CurrentDate$+$1)
What is the intensity of walking? T&S:$Answer(e.intensity) \wedge e.type ==$ Exercise $\wedge\, e.kind ==$ Walking
$Click(e) \wedge e.type ==$ HypoEvent $\wedge\, e.time ==$ 19:54pm $Click(e) \wedge e.type ==$ HypoAction $\wedge\, e.time ==$ 19:54pm Does she usually just take sugar? T:$Answer(Cond(e.type ==$ HypoAction $=> e.food ==$ Sugar$))$ S:$Answer(Count(e.food ==$ Sugar $\wedge\, e.type ==$ BGL$))$

VII. Conclusion

We introduced a new question answering task where users can express their information needs through a combination of natural language questions and direct actions within a graphical user interface. We created a dataset of real interactions between physicians and a GUI tool showing patient data, and designed a procedure for generating a much larger dataset of artificial interactions that aims to preserve the major characteristics of the real data. Experimental evaluations show that a new sequence generation architecture that models context dependency through multiple attention levels and copy mechanisms significantly outperforms traditional sequence generation baselines.

Acknowledgments

This work was partly supported by grant 1R21EB022356 from the National Institutes of Health. We would like to thank Dr. Frank Schwartz for contributing real interactions with PhysioGraph, Quintin Fettes and Yi Yu for their help with recording and pre-processing the interactions, and Robin Kelby for annotating logical forms. We would also like to thank the anonymous reviewers for their useful comments.

References

[1] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, "Towards AI-complete question answering: A set of prerequisite toy tasks," *International Conference on Learning Representations*, pp. 1–15, 2016.

[2] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *International Conference on Learning Representations*, pp. 1–15, 2015.

[4] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[5] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 1631–1640, 2016.

[6] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *International Conference on Learning Representations*, pp. 1–15, 2016.

[7] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, pp. 1–15, 2015.